TUGAS AKHIR – TM 141585

PERANCANGAN SISTEM KENDALI *TRACKING OBJECT* BERBASIS *DYNAMIC WAYPOINT* PADA QUADCOPTER

DILI KURNIAWAN PUTRA
NRP 2114 105 021

Dosen Pembimbing
Ir. Bambang Pramujati, M.Eng., Ph.D

PROGRAM SARJANA
LABORATORIUM OTOMASI INDUSTRI
JURUSAN TEKNIK MESIN
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

FINAL PROJECT – TM 141585

DESIGN OF TRACKING OBJECT CONTROL SYSTEM BASED ON DYNAMIC WAYPOINT ON QUADCOPTER

DILI KURNIAWAN PUTRA
NRP 2114 105 021

Academic Advisor
Ir. Bambang Pramujati, M.Eng., Ph.D

BACHELOR DEGREE PROGRAM
INDUSTRIAL AUTOMATION LABORATORY
MECHANICAL ENGINEERING DEPARTMENT
INDUSTRIAL TECHNOLOGY FACULTY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA 2016

LEMBAR PENGESAHAN

PERANCANGAN SISTEM KENDALI *TRACKING OBJECT* BERBASIS *DYNAMIC WAYPOINT* PADA QUADCOPTER

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
Pada
Program Studi S-1 Jurusan Teknik Mesin
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

Dili Kurniawan Putra
NRP : 2114 105 021

Disetujui oleh Tim Penguji Tugas Akhir:

1. Ir.Bambang Pramujati Msc.Eng, PhD
   (NIP. 196912031994031001) .....(Pembimbing)
2. Dr.Eng Unggul Wasiwitono, ST, M.Eng.Sc
   (NIP. 197805102001121001) .....(Penguji 1)
3. Ari Kurniawan Saputra, ST, MT
   (NIP. 198604012015041001) .....(Penguji 2)
4. Latifah Nurahmi,ST.,M.Sc, Ph.D
   (NIP. 210000011) .....(Penguji 3)

SURABAYA 2016

PERANCANGAN SISTEM KENDALI *TRACKING OBJECT* BERBASIS *DYNAMIC WAYPOINT* PADA QUADCOPTER

Nama Mahasiswa : Dili Kurniawan Putra
NRP : 2114105021
Jurusan : Teknik Mesin
Dosen Pembimbing : Ir. Bambang Pramujati, M.Eng., Ph.D

Abstrak

UAV (Unmanned Aerial Vehicle) adalah wahana tanpa awak yang memiliki sistem kendali jarak jauh menggunakan remote control ataupun ground station. UAV digunakan untuk berbagai keperluan antara lain pengintaian, surveillance, pengambilan gambar/video (Aerialphotography) dan lain-lain. UAV dengan tipe multirotor memiliki banyak varian antara lain tricopter, quadcopter, hexacopter dan octocopter. Perbedaan mendasar tipe multirotor tersebut adalah jumlah lengan motor. Quadcopter memiliki 4 lengan yang masing-masing terintegrasi dengan motor dan propeller.

Pada penelitian ini, dirancang quadcopter dengan kendali tracking object menggunakan GPS (Global Positioning System) sebagai alat untuk memberikan data lokasi dari obyek. Perancangan dimulai dengan mendesain quadcopter menggunakan software Solidworks untuk mendapatkan data fisik quadcopter seperti massa dan inersia. Simulasi open loop dilakukan dengan menggunakan sofware Matlab. Pada proses simulasi, pemodelan matematis quadcopter dibuat linier dengan pendekatan Single Input Single Output. Setelah mendapatkan respon transien pada sistem open loop, proses simulasi dilanjutkan dengan memberikan sistem kendali PID (Proportional Integral Derivative) pada masing-masing pemodelan gerak pitch, roll, yaw,

altittude, longitudinal dan lateral. Kendali PD digunakan dalam proses simulasi.

Proses simulasi pada gerak utama (roll, pitch , yaw, dan gerak vertikal) menghasilkan respon transien sesuai spesifikasi desain yaitu settling time < 2 detik dan overshoot < 30%. Hasil simulasi pada gerak longitudinal dan lateral menghasilkan respon transien sesuai spesifikasi desain yaitu settling time < 2 detik dan overshoot < 50%. Hasil pengujian dalam perancangan quadcopter menghasilkan respon gerak aktual pada masing-masing sistem mengikuti sinyal referensi/setpoint.

Kata Kunci : Quadcopter; UAV; Controller; PID; Stabil

DESIGN OF TRACKING OBJECT CONTROL SYSTEM BASED ON DYNAMIC WAYPOINT ON QUADCOPTER

Name	**: Dili Kurniawan Putra**
NRP	**: 2114105021**
Department	**: Teknik Mesin**
Academic Advisor	**: Ir. Bambang Pramujati, M.Eng., Ph.D**

Abstract

UAV (Unmanned Aerial Vehicle) is unmanned vehicle that has long-range control system by using either remote control or ground station. UAV can be used for various purposes, such as, reconnaissance, surveillance, video recording and aerial photography. There are several types of UAV multirotor, such as tri-copter, quad-copter, hexa-copter and octo-copter. The main difference among those type of multirotor is the number of motor arm. Quadcopter has 4 arms which each of the arm are integrated with motor and propeller.

On this research, a quadcopter is developed having tracking object controller by using GPS as equipment to hand-over current data location from the object. Moreover in the designation, I started with designing the quadcopter by using software called, Solidworks. This software is used for obtaining physical data of the quadcopter like mass and inertia. Open loop simulation executed by using software named, Matlab. In simulation process, the mathematical model are made linearly with single input single output approach. Next, after receiving a transient's response in the open loop system, simulation process was continued by giving PID controller to each of them pitch, roll, yaw, altitude, longitudinal, and lateral motion. PD compensator use while simulation.

Further, while having simulation process in the main motion (roll, pitch, yaw and vertical motion), the results of transient

response based on design specification is settling time < 2 seconds and overshoot < 30%. Simulation result on longitudinal and lateral motion produce transient response according to the design of specification itself, that is, settling time < 2 seconds and overshoot < 50%. The result of testing and researching quadcopter is creating an actual motion response of each system was following reference signal/set-point.

Keywords : Quadcopter; UAV; Controller; PID; Stable

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SIMBOL

ϕ : Sudut Roll (^{o})
θ : Sudut Pitch (^{o})
ψ : Sudut Yaw (^{o})
F_t : Gaya *Thrust* (N)
M_t : Momen Torsi (Nm)
K_M : Konstanta Momen
K_f : Konstanta Drag
v : Tegangan (v)
K_{mot} : Konstanta motor
R_{mot} : Hambatan listrik pada motor (ohm)
Ω_{hover} : Putaran motor kondisi *hover* (rad/s)
v_{hover} : Input tegangan motor yang dibutuhkan pada kondisi *hover*
v_{bat} : Tegangan baterai
Ω_i : Putaran motor (rad/s)
K_p : Konstanta *Proportional*
K_I : Konstanta integral
K_D : Konstanta derivatif
θ_{zc} : Sudut zeros tambahan
θ_p : Sudut poles
L : *Delay time*
T : *Time constant*
K_{cr} : *Critical gain*
T_i : *Integral time*
T_D : *Derivative time*

BAB I
PENDAHULUAN

1.1 Latar Belakang

UAV (*Unmanned Aerial Vehicle)* adalah wahana tanpa awak yang memiliki sistem kendali jarak jauh menggunakan *remote control* ataupun *ground station*. UAV digunakan untuk berbagai misi tertentu, antara lain:

a. **Pengintaian jarak jauh:** UAV dilengkapi dengan kamera pengintai yang dapat mengawasi keberadaan suatu target tertentu dan mengamati lingkungan sekitar.
b. **Pencarian:** UAV dilengkapi dengan kamera. Kamera tersebut didukung dengan pengolahan citra (*image processing*) yang digunakan untuk mencari obyek tertentu. Kriteria obyek dideskripsikan oleh perangkat pengolahan citra berdasarkan properti nya seperti dimensi, warna, temperatur, dan lain-lain.
c. **Alutsista:** UAV dilengkapi oleh persenjataan serta amunisi.
d. ***Aerial Photography*:** UAV dikontrol oleh *user* untuk mengambil gambar/video. Video tersebut biasanya digunakan untuk kebutuhan *entertainment* maupun visual gambar tentang suatu berita dimedan yang sulit.

UAV memiliki tipe yang beragam, salah satunya adalah multirotor. Multirotor merupakan UAV berpendorong baling baling (*blade*) lebih dari satu. Uav tipe multirotor memiliki bentuk dan konfigurasi motor yang bermacam macam, antara lain:

1. Tricopter

Tricopter memiliki 3 lengan (*arm)* dan memiliki bentuk menyerupai huruf "Y". Konfigurasi motor pada tricopter adalah:

- Y3: Tricopter jenis ini menggunakan 1 motor pada setiap lenganya.
- Y6: Tricopter jenis Y6 menggunakan 2 motor yang letaknya berkebalikan antara satu dengan lainya.

2. Quadcopter

Quadcopter adalah UAV multirotor yang memiliki 4 lengan (*arm*). Setiap lengan quadcopter memiliki 1 motor dan baling-baling (*blade*). Quadcopter memiliki dua tipe umum yaitu:

- *X configuration*

  Quadcopter tipe-X memiliki bentuk menyerupai huruf "X". Sisi depan pada quadcopter tidak sejajar dengan lengan (*arm*), melainkan menyimpang 45 derajat.

- *Plus Configuration*

  Quadcopter tipe plus memiliki bentuk menyerupai tanda "+". Sisi depan quadcopter sejajar dan atau tegak lurus terhadap masing-masing lengan.

3. Hexacopter

Hexacopter adalah UAV multirotor yang memiliki 6 lengan. Masing masing lengan hexacopter memiliki satu pasang motor dan propeller. Hexacopter biasa digunakan untuk keperluan *aerial photography* karena memiliki kemampuan membawa beban yang besar seperti kamera DSLR.

4. Octocopter

Octocopter merupakan UAV multirotor dengan dimensi yang cukup besar karena memiliki 8 lengan. Octocopter mampu menghadapi gangguan luar seperti hembusan angin. Dengan kemampuanya tersebut, octocopter dapat dioperasikan pada lingkungan dengan angin kencang seperti kawasan bibir pantai dan dataran tinggi.

Multirotor dirancang berdasarkan medan yang dihadapi dan juga kegunaanya. Secara umum, UAV multirotor tipe quadcopter merupakan tipe standar yang memiliki kegunaan yang beragam dan mampu dioperasikan dalam kondisi lingkungan tenang maupun berangin kencang. Keuntungan dari quadcopter dibanding dengan tipe lain adalah unggul dalam *maneuverability* dan dapat meningkatkan kemampuan membawa beban [9]. Pada gambar 1.1

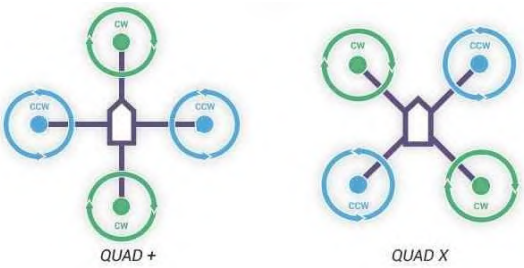
memperlihatkan perbandingan beberapa tipe multirotor yang sering digunakan.

Categories	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
Power Cost	2	2	2	2	1	4	3	3
Control Cost	1	1	4	2	3	3	2	1
Payload/volume	2	2	4	3	3	1	2	1
Maneuverability	4	2	2	3	3	1	3	3
Mechanics Simplicity	1	3	3	1	4	4	1	1
Aerodynamics Complexity	1	1	1	1	4	3	1	1
Low Speed Flight	4	3	4	3	4	4	2	2
High Speed Flight	2	4	1	2	3	1	3	3
Miniaturization	2	3	4	2	3	1	2	4
Survivability	1	3	3	1	1	3	2	3
Stationary Flight	4	4	4	4	4	3	1	2
TOTAL	24	28	32	24	33	28	22	24

A=Single Rotor, B=Axial Rotor, C=Coaxial Rotor, D=Tandem Rotors, E=Quadrotor, F=Blimp, G=Bird-like, H=Insect-like. 1=Poor, 4=Excellent

Gambar 1.1 Perbandingan Beberapa Tipe Multirotor [9]

Dari gambar 1.1 terlihat bahwa quadcopter memiliki perolehan nilai yang tinggi. Dengan kata lain, quadcopter memiliki kemampuan yang lebih baik daripada tipe multirotor lainya. Berbeda dengan tipe lainya, quadcopter dapat dibuat dengan dimensi besar maupun kecil sesuai dengan kebutuhan yang diinginkan. Bentuk dan konfigurasi motor pada quadcopter ditunjukan pada gambar 1.2.



Gambar 1.2 Konfigurasi Rotor pada UAV Tipe Quadcopter

Dalam memenuhi fungsinya, quadcopter harus memiliki kemampuan untuk menghadapi ganggunan dari lingkungan sekitar. Oleh karena itu, sistem kendali berperan penting untuk menjaga quadcopter tetap stabil. Salah satu masalah yang dihadapi ketika mengoperasikan quadcopter adalah kesalahan pengguna (*human error*). Hal itu terjadi karena kurangnya kemampuan dan pengalaman dalam menerbangkan UAV. Bila tidak didukung dengan kamera FPV (*First Person View*) maka user akan mengalami kesulitan untuk mengetahui orientasi quadcopter ketika beroperasi dengan radius jelajah yang luas. Untuk mencegah terjadinya *crash* akibat *human error*, maka dibutuhkan adanya sistem *autonomous*. Salah satu sistem *autonomous* yang diaplikasikan pada UAV adalah metode *tracking object*. Keunggulan dari metode *tracking object* adalah mengantisipasi adanya kegagalan kontrol pada *user* serta mampu melakukan *tracking* pada *object* yang menjadi *target tracking*.

Terdapat beberapa penelitian dalam pengembangan sistem *autonomous* pada UAV. Bouabdallah (2004) melakukan analisa mengenai kestabilan posisi pada pemodelan *non-linear* quadcopter tipe plus menggunakan teori lyapunov dan menerapkan hasil analisanya kedalam perancangan. Pada penelitian tersebut didapatkan hasil simulasi yaitu nilai settling time 1,4 detik untuk kestabilan posisi. Heba (2014) melakukan analisa kestabilan posisi pada model *non-linear* quadcopter tipe plus menggunakan kendali PID. Parameter P, I, dan D diperoleh dengan menggunakan *genetic algorithm* yang tersedia pada *software* MATLAB. Denys (2012) melakukan analisa *position tracking control* dengan memanfaatkan pengolahan citra (*image processing*) untuk mengestimasikan posisi objek relatif terhadap posisi quadcopter.

1.2 Rumusan Masalah

Perumusan masalah pada tugas akhir ini adalah bagaimana memodelkan dan menganalisa kestabilan sistem kendali *tracking object* pada quadcopter dengan menggunakan metode kontrol PID serta mendapatkan parameter P, I dan D yang sesuai.

1.3 Tujuan Tugas Akhir

Tujuan yang ingin dicapai pada penelitian ini adalah mendapatkan model linear dan menganalisa kestabilan sistem kendali *tracking object* serta mendapatkan parameter PID yang sesuai untuk quadcopter.

1.4 Batasan Masalah

1. Analisa quadcopter tipe *X-configuration* dengan panjang diagonal antara rotor sebesar 450 mm.
2. Konstruksi rangka pada quadcopter dianggap *rigid* (kaku).
3. Gaya *drag* oleh lingkungan diabaikan.
4. Simulasi diterapkan pada model *non-linear* quadcopter yang telah dilinearisasi pada kondisi *hover*.
5. Pendekatan *single input single output* pada proses simulasi.

1.5 Manfaat Tugas Akhir

Manfaat yang dihasilkan dalam tugas akhir ini adalah:

1. Mendapatkan pengetahuan tentang sistem kendali *tracking object* pada quadcopter.
2. Hasil penelitian ini dapat digunakan sebagai acuan dalam merancang quadcopter untuk. kegiatan *surveillance*.
3. Penelitian ini dapat diimplementasikan untuk mengantisipasi kegagalan/*crash* dalam mengoperasikan quadcopter.
4. Sebagai alat untuk kegiatan *aerial photography* pada medan yang sulit.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA & DASAR TEORI

2.1 Penelitian Terdahulu

Sistem kendali adalah suatu sistem yang terdiri dari subsistem dan *plants* yang bertujuan untuk mendapatkan *output* sesuai dengan yang diinginkan, dengan kata lain untuk mengendalikan *output*. Sistem kendali dibangun dengan empat alasan utama yaitu perluasan kemampuan, kendali jarak jauh, kenyamanan bentuk masukan, dan memperkecil gangguan. Sistem kendali dibagi menjadi dua jenis yaitu *open loop* dan *close loop*. Pada *open loop system*, nilai *output* dan nilai *input* tidak dibandingkan. Sementara pada *close loop system* merupakan kebalikan dari *open loop system*, sehingga dapat dikatakan nilai error dari *input* dan *output* menjadi masukan bagi sistem kendali.

Heba (2014) melakukan perencanaan sistem kendali posisi quadcopter dengan pemodelan *non-linear*. Pemodelan non-*linear* dibagi menjadi 6 subsistem yaitu *pitch,roll yaw,altitude,* X, dan Y. Masing-masing subsistem diberi kendali proporsional dan derivatif dengan parameter Kp dan Kd didapat menggunakan *genetic algorithm* yang tersedia pada *software* MATLAB. Dari hasil simulasi didapatkan parameter $\{Kp,Kd\}_{Altitude}$ = {5.2,1.3}, $\{Kp,Kd\}_{Attitude}$ = {4.5,0.5}, $\{Kp,Kd\}_{Heading}$ = {3.9,0.7} dan $\{Kp,Kd\}_{Posisi}$ = {7.5,4.2}.

Penelitian tentang multirotor dilakukan oleh Bohdanov (2012) tentang perbandingan pemodelan *non-linear* terhadap pemodelan *linear* pada sistem dinamis multirotor. Dari hasil yang didapatkan, pemodelan *linear* dianggap dapat mewakili pemodelan non-*linear*. Kendali PID digunakan untuk menstabilkan gerak rotasi (*roll*, *pitch*, *yaw*) dan ketinggian. Kendali LQG digunakan untuk mengontrol gerak translasi quadcopter pada sumbu x dan z. Kendali posisi dirancang menggunakan *image processing* dan kalman filter untuk mengestimasikan gerakan translasi multirotor.

Marcelo (2011) melakukan simulasi pemodelan *linear* quadcopter dengan acuan *Single Input Single Output*. Kendali PID

digunakan untuk mengendalikan gerak *pitch,roll,yaw*, dan posisi (x dan y). Fungsi transfer motor diidentifikasi berdasarkan hasil eksperimen dengan asumsi *first order system.* Proses pengambilan data dilakukan menggunakan microcontroller dan sensor putaran. Data tersebut diolah menggunakan *system identification tool* pada Matlab. Simulasi meghasilkan nilai *settling time* = 3.5 detik pada sistem kendali posisi (x dan z).

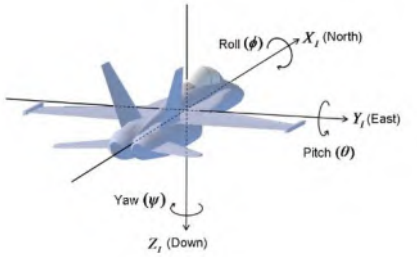
2.2 Dasar Teori

2.2.1 Sistem Koordinat

Untuk mendapatkan persamaan gerak quadcopter relatif terhadap daratan, maka persamaan gerak quadcopter akan berhubungan langsung dengan perubahan sistem koordinat. Sistem koordinat yang bekerja pada quadcopter antara lain *inertial frame coordinates, vehicle-1 frame, vehicle-2 frame,* dan *body frame.* Masing-masing sistem koordinat dijabarkan dalam subbab ini.

2.2.1.1 *Inertial Frame*

Inertial frame merupakan suatu sistem koordinat bumi (*Inertial Frame Coordinates*) dimana koordinat ini tidak berotasi dan bergerak. Sumbu yang terdapat pada *inertial frame* adalah seperti pada gambar 2.1.



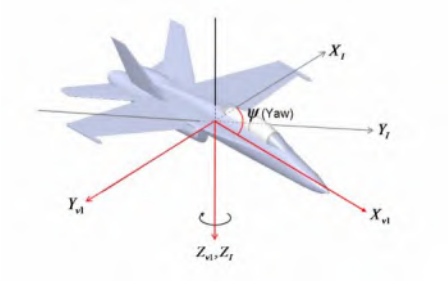
Gambar 2.1 *Earth-Fixed Coordinate* (*Inertial frame Coordinates)*

Dari gambar 2.1 menjelaskan masing-masing sumbu *inertial frame* yaitu X (Utara), Y (Timur) dan Z (mengarah ke daratan)

dengan notasi $\{X_I, Y_I, Z_I\}$. Sistem koordinat bumi merupakan sistem koordinat absolut lokasi dan orientasi quadcopter terhadap daratan. Orientasi *attitude*, *heading* dan *position* quadcopter ditentukan berdasarkan referensi sumbu Timur, Utara dan daratan.

2.2.1.2 *Vehicle-1 Frame (Yaw Rotation)*

Yawing merupakan suatu perubahan rotasi sudut terhadap sumbu Z pada *inertial frame coordinates*. Rotasi sudut *yaw* menyebabkan adanya koordinat baru. Sumbu Z berhimbit dan sejajar dengan *inertial frame*. Sumbu X dan Y yang baru memiliki perbedaan sudut sebesar ψ.

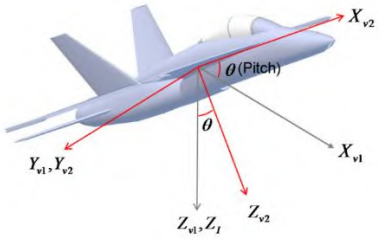


Gambar 2.2 *Vehicle-1 frame*

Komponen koordinat *Vehicle-1 frame* dinotasikan dengan$\{X_{v1}, Y_{v1}, Z_{v1}\}$. *Vehicle-1 frame* merupakan sistem koordinat baru hasil dari rotasi sebesar, ψ. Sistem koordinat tersebut diilustrasikan pada gambar 2.2 dengan garis warna merah.

2.2.1.3 *Vehicle-2 Frame*

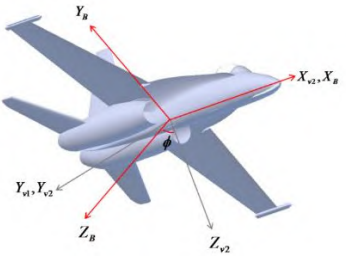
Pitch merupakan rotasi terhadap sumbu Y sebesar θ relatif terhadap koordinat *vehicle-1 frame*. Seperti yang terlihat pada gambar 2.3, garis berwarna abu-abu merupakan *vehicle-1 frame* sedangkan warna merah merupaan *vehicle-2 frame* dengan notasi $\{X_{v2}, Y_{v2}, Z_{v2}\}$. Sumbu y pada *vehicle-1 frame* dan *vehicle-2 frame* berhimpit, dan koordinat X dan Z pada pada $v2$ memiliki perbedaan sudut sebesar θ terhadap *vehicle-1 frame*.



Gambar 2.3 *Vehicle-2 Frame*

2.2.1.4 *Body Frame*

Body frame merupakan sistem koordinat pada *body* quadcopter dengan notasi $\{X_B, Y_B, Z_B\}$ seperti pada gambar 2.4. Muka depan (*head*) quadcopter sejajar dan searah dengan sumbu X positif pada *body frame coordinates*. Sayap kanan quadcopter sejajar dengan sumbu y positif *body frame* dan sumbu z positif *body frame* sejajar dengan sisi bawah (*fuselage*) quadcopter.



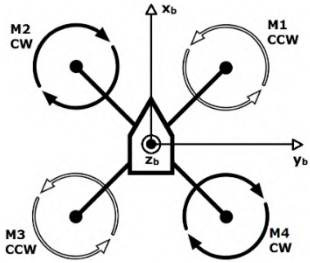
Gambar 2.4 *Body Frame Coordinates*

2.2.2 Konsep Dasar Quadcopter (*X-Configuration*)

Desain quadcopter terdiri dari 4 lengan yang masing-masing memiliki satu buah motor. Posisi motor yang ada pada quadcopter secara berturut-turut adalah M1 (Depan-Kanan), M2 (Depan-Kiri), M3 (Belakang-Kiri) dan M4 (Belakang-Kanan). Bentuk quadcopter tipe X dapat simetris maupun tidak simetris tergantung dengan kebutuhan. Bentuk yang tidak simetris cenderung memiliki

keuntungan dalam segi tata letak komponen. Bentuk yang simetris memiliki keuntungan yaitu proses perakitanya lebih mudah.

Masing-masing motor pada quadcopter memiliki baling-baling (*propeller*) berukuran sama namun berputar dengan arah putaran yang berbeda. Seperti terlihat pada Gambar 2.5, motor 1 dan 3 berputar berlawanan arah jarum jam (CCW) sedangkan motor 2 dan 4 berputar searah dengan jarum jam (CW). Hal itu dilakukan untuk mengantisipasi adanya efek giroskopik.



Gambar 2.5 Konfigurasi Motor pada Quadcopter Tipe *X-Configuration*

Quadcopter merupakan sistem dinamis yang memiliki 6 persamaan gerak yaitu $(z, \phi, \theta, \psi, y, x)$. Meskipun memiliki 6 persamaan, quadcopter adalah sistem dinamis dengan 4 derajat kebebasan (4DOF) yaitu (z, ϕ, θ, ψ). x, y, z merepresentasikan gerak translasi pusat masa quadcopter pada *body frame* relatif terhadap sumbu koordinat bumi. Sedangkan ϕ, θ, ψ merepresentasikan perbedaan sudut quadcopter yang berputar pada sumbu x,y, dan z relatif terhadap sumbu koordinat bumi. Gerakan quadcopter yang menyebabkan adanya perubahan sudut ϕ, disebut *pitch* sedangkan gerakan quadcopter yang menyebabkan perubahan sudut θ, disebut *roll*. Gerakan *pitch* dan *roll* dikategorikan sebagai *attitude* dari quadcopter. Gerakan quadcopter yang menyebabkan perubahan sudut ψ, disebut *yaw*. Gerakan *yaw* relatif terhadap koordinat bumi (*inertial frame*) dikategorikan sebagai *heading*. Jarak vertikal quadcopter terhadap bumi, z disebut *altitude* sedangkan titik koordinat x dan y

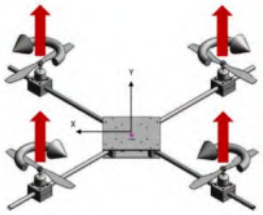
dikategorikan sebagai posisi quadcopter. Quadcopter akan bergerak translasi pada sumbu x ketika terjadi *rolling* dan akan bergerak translasi pada sumbu y ketika terjadi *pitching*.

2.2.3 Metode Gerak Utama pada Quadcopter

Beberapa pergerakan dasar yang memungkinkan quadcopter dapat mencapai ketinggian (*altitude)* dan sikap *(attitude)* tertentu. Pada gambar-gambar di bawah ini panah warna merah menunjukkan nilai gaya *thrust* yang besar/putaran motor paling tinggi dan panah warna hijau menunjukan nilai gaya *thrust* yang kecil/putaran motor paling rendah pada masing-masing motor.

2.2.3.1 Hovering

Hovering merupakan kondisi quadcopter ketika dalam keadaan melayang. Hal itu terjadi karena adanya gaya *thrust* yang dihasilkan masing-masing motor sama besar dan menghasilkan total gaya angkat yang sama dengan berat quadcopter itu sendiri.



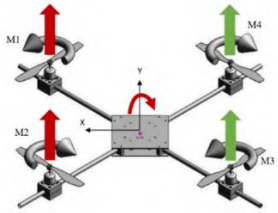
Gambar 2.6 Putaran Motor Ketika *Hovering*

Ketika *hovering*, quadcopter akan mengalami keseimbangan secara translasi maupun rotasi. Gambar 2.6 menunjukan gaya *thrust* yang dihasilkan masing-masing motor.

2.2.3.2 *Pitching*

Pitching merupakan gerak rotasi quadcopter terhadap sumbu horizontal y. Kondisi *pitching* terjadi ketika kecepatan putaran motor 1-2 berbeda dengan 3-4. Hal itu membuat adanya perbedaan momen terhadap sumbu y dan quadcopter akan berotasi kearah

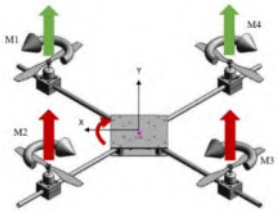
putaran motor yang lebih rendah seperti yang diilustrasikan pada gambar 2.7.



Gambar 2.7 Putaran Motor Ketika *Pitching*

2.2.3.2 Rolling

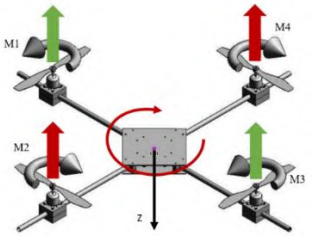
Gerakan *rolling* adalah gerakan akibat adanya perubahan momen pada sumbu horizontal x. Hal itu terjadi ketika ada perbedaan kecepatan putaran motor 1-4 dengan 3-2. Quadcopter akan *rolling* kearah momen yang lebih kecil pada sumbu horizontal x seperti diilustrasikan pada gambar 2.8.



Gambar 2.8 Putaran Motor Ketika *Rolling*

2.2.3.4 Yawing

Yawing merupakan suatu efek giroskopik karena adanya perbedaan kecepatan putaran pasangan motor 1-3 dengan 2-4. Perbedaan putaran kedua pasangan motor tersebut menyebabkan torsi yang memutar quadcopter pada sumbu z. Pada gambar 2.9 terlihat bahwa motor 2 dan 4 memiliki putaran yang lebih tinggi dibandingkan motor 1 dan 3 sehingga menimbulkan momen torsi yang memutar quadcopter pada sumbu z.



Gambar 2.9 Putaran Motor Ketika *Yawing*

2.2.4 Kinematika Model Quadcopter

Kinematika quadcopter dibagi menjadi dua bagian yaitu rotasi dan translasi. Gerakan rotasi didapatkan menggunakan persamaan rotasi matrix. Gerak translasi didapat dari persamaan yang berhubungan dengan hukum Newton.

2.2.4.1 Persamaan Rotasi Matrix (Euler)

Sistem koordinat quadcopter sangat berpengaruh untuk menentukan orientasi quadcopter. Sistem koordinat yang berlaku untuk menjelaskan pergerakan quadcopter adalah:

1. *Inertial frame*
2. *Body frame*
3. *Vehicle-1 frame*
4. *Vehicle-2 frame*

Inertial frame diputar terhadap sumbu Z sebesar sudut *yaw* (ψ), sehingga menghasilkan koordinat baru yaitu *vehicle-1 frame*. Transformasi *inertial frame*,i ke *vehicle-1 frame*,$v1$ dapat dituliskan kedalam vektor matrix yaitu:

$$R_i^{v1} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Notasi R_i^{v1} pada persamaan 2.1 merupakan rotasi dari *inertial frame* menuju *vehicle-1 frame*. Setelah mendapatkan koordinat

baru karena adanya vector rotasi R_i^{v1}, sistem koordinat tersebut diputar terhadap sumbu y sebesar, θ dan menghasilkan sistem koordinat baru yaitu *vehicle-2 frame*. *Vehicle-2 frame* dapat ditulis sebagai matrix rotasi seperti pada persamaan 2.2.

$$R_{v1}^{v2} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.2}$$

Rotasi terakhir adalah ketika sistem koordinat relatif R_{v1}^{v2} diputar terhadap sumbu x sebesar,$\emptyset$ dan akan menghasilkan rotasi matrix relatif terhadap *vehicle-2 frame* yaitu:

$$R_{v2}^{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2.3}$$

Untuk mendapatkan rotasi matrix hubungan gerak rotasi dan translasi pada *body frame coordinates* terhadap *inertial frame coordinates*, maka dilakukan subtitusi persamaan 2.1,2.2, dan 2.3. Rotasi matriks merupakan invers dari gabungan persamaan rotasi matrix masing-masing sistem koordinat seperti pada persamaan 2.4.

$$R_b^i = \left[R_{v2}^b \, R_{v1}^{v2} \, R_i^{v1}\right]^T$$

$$R_b^i = \begin{bmatrix} \mathrm{c}\,\theta\,\mathrm{c}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{c}\,\psi + \mathrm{s}\,\phi\,\mathrm{s}\,\psi \\ \mathrm{c}\,\theta\,\mathrm{s}\,\psi & \mathrm{s}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi + \mathrm{c}\,\theta\,\mathrm{c}\,\psi & \mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{s}\,\psi - \mathrm{s}\,\theta\,\mathrm{s}\,\psi \\ -\mathrm{s}\,\theta & \mathrm{c}\,\phi\,\mathrm{s}\,\theta & \mathrm{c}\,\phi\,\mathrm{c}\,\theta \end{bmatrix} \tag{2.4}$$

2.2.4.2 Persamaan *Euler Rates*

Hubungan antara $\dot{\phi}$, $\dot{\theta}$ dan $\dot{\psi}$ dengan p, q, r dimana kecepatan perubahan sudut pada *body frame coordinates* diubah menjadi sistem koordinat bumi (*body frame coordinates*) quadcopter ditunjukkan pada persamaan 2.5 dan 2.6.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = E \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.5}$$

$$E = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\theta & s\phi c\theta \end{bmatrix} \quad (2.6)$$

Dari persamaan 2.6, didapatkan persamaan perubahan gerak berdasarkan rotasi quadcopter yaitu seperti pada persamaan 2.7.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = E^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.7)$$

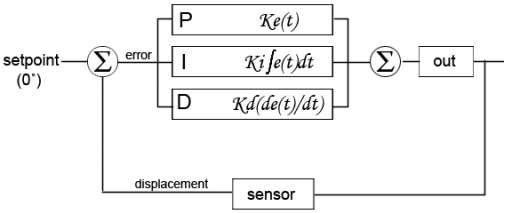
2.2.5 Kendali PID (*Proportional-Integral-Derivative*)

Sistem kendali yang ditambahkan dengan PID saat ini sangat popular. Hal ini karena struktur dari pengendalian ini sangat sederhana. Metode yang digunakan untuk simulasi PID ini adalah metode *trial error*. Kendali PID merupakan sistem kendali yang menggabungkan antara tiga macam kendali yaitu *proportional*, *integral* dan *derivatif*. Penggabungan ketiga macam kendali dan pemilihan konstanta yang tepat dapat menutupi kekurangan dan menonjolkan kelebihan masing-masing pengendalian. Misalnya, kendali *proportional* cukup mampu untuk memperbaiki *rise time* dan *settling time* namun meninggalkan *offset*. Kekurangan ini dapat ditutupi dengan cara menggabungkan dengan kendali integral yang dapat menghilangkan *offset* dan juga mengurangi terjadinya *overshoot* yang terlalu luas, serta mampu menghilagkan *steady state error*. Akan tetapi, kendali integral dapat menyebabkan respon sistem menjadi lambat. Penanggulangan respon sistem yang lambat menggunakan kendali derivatif. Skema blok diagram sistem kendali PID dapat dilihat pada gambar 2.10. Karakteristik Kendali PID sangat dipengaruhi oleh nilai P, I dan D, penyetelan dari konstanta Kp, Ki, dan Kd harus terus menerus diatur kembali (*trial-error)* untuk mendapatkan performa yang baik.

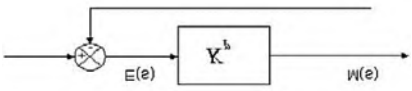
2.2.5.1 Pengaruh Sistem Kendali Proporsional

Kendali proporsional memiliki keluaran yang sebanding (*proportional*) dengan besarnya sinyal kesalahan (*error*). Secara

umum, kendali proporsional memiliki keluaran sebesar perkalian antara gain Kp dengan *error*. Skema sistem kendali proporsional ditunjukan pada gambar 2.11.



Gambar 2.10 Blok Diagram Kendali PID



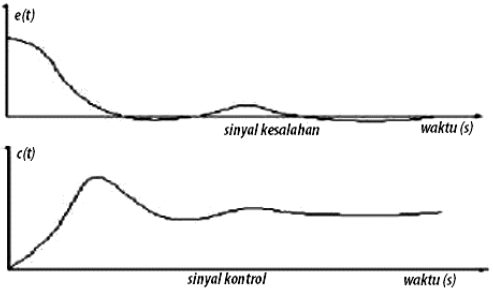
Gambar 2.11 Kendali Proporsional

Dari skema gambar 2.11, dapat terlihat bahwa semakin besar nilai Kp, maka keluaran kendali proporsional semakin besar begitu juga sebaliknya. Kendali proporsional adalah parameter utama dalam mengendalikan quadcopter dan sangat berpengaruh pada kestabilan quadcopter. Nilai *output* kendali proporsional akan menjadi input sistem dinamis quadcopter. Bila nilai Kp terlalu besar, maka input sistem dinamis quadcopter akan besar. Hal itu menyebabkan munculnya osilasi pada gerakan quadcopter. Bila Kp terlalu kecil, quadcopter tidak merespon adanya perubahan *input* secara cepat.

2.2.5.2 Pengaruh Sistem Kendali Integral

Kendali integral digunakan ketika suatu respon sistem memiliki *steady state error*. Prinsip dasar dari kendali integral adalah menjumlahkan/integrasi sinyal kesalahan (*error*) terhadap waktu. Hasil dari integrasi dikali dengan kontanta integral, Ki. *Output* kendali integral akan terus bertambah terhadap waktu sampai sistem mencapai *setpoint*. Sinyal keluaran kendali integral

merupakan hasil luasan grafik hubungan sinyal kesalahan terhadap waktu. Grafik pada gambar 2.12 merupakan hubungan sinyal kesalahan dengan sinyal keluaran sisteem kendali integral.

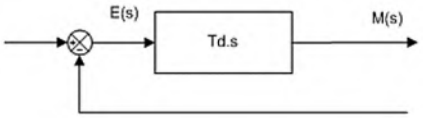


Gambar 2.12. Grafik Sinyal Kesalahan dan Sinyal Kontrol

2.2.5.3 Pengaruh Sistem Kendali Derivatif

Sistem kendali derivatif memberi sinyal keluaran seperti hal nya operasi differensial. Kendali dervatif akan mengevaluasi perubahan sinyal kesalahan terhadap waktu. Sinyal keluaran kendali derivatif akan bernilai nol apabila tidak ada perubahan sinyal kesalahan terhadap waktu. Blok diagram sistem kendali derivatif ditunjukan pada gambar 2.13.

Karakter kendali derivatif akan meminimalisir adanya *overshoot* pada respon sistem. Nilai Kd yang terlalu besar akan menghambat respon sistem menuju *setpoint*. Nilai Kd terlalu kecil menyebabkan sistem mengalami *overshoot*.



Gambar 2.13. Sistem Kendali Derivatif

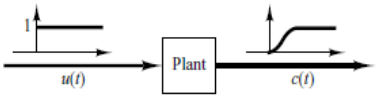
2.2.6 Metode Tuning PID

Terdapat beberapa cara yang biasa digunakan untuk melakukan *tuning* parameter PID, diantaranya yang sering

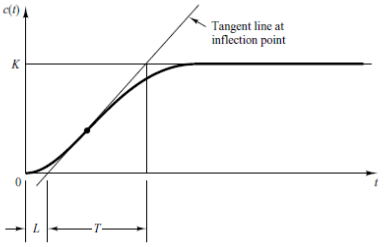
digunakan adalah *trial error* dan metode ziegler-nichols. *Trial error* biasa digunakan untuk memperbaiki respon transien suatu sistem sedangkan metode ziegler- nichols digunakan sebagai tahap awal sebelum proses *trial error* dilakukan.

- **Metode Ziegler-Nichols**

Salah satu metode tuning kontrol PID yang umum digunakan adalah metode *Ziegler-Nichols*. Metode ini terbagi menjadi 2 metode, yaitu metode pertama untuk sistem dengan karakteristik *first order system* dan *second order system*, dan metode kedua untuk sistem dengan karakteristik *third order system* dan selanjutnya. Pada metode pertama, Ziegler dan Nichols mengajukan aturan untuk menentukan nilai *Proportional Gain* (K_P), *Integral Time*, T_i dan *Derivative Time*, T_D berdasarkan karakteristik respon transien dari sistem untuk *delay time (L)* dan *time constant(T).*



Gambar 2.14. Plant Dengan Unit Step Response



Gambar 2.15 *Delay Time (L)* dan *Time Constant (Tc)*

Gambar 2.14 dan 2.15 menunjukkan blok diagram *first order system* dan penentuan *delay time (L)* dan *time constant (T)* pada suatu respon *transient* sistem. *Delay time* dan *Time Constant*

direpresentasikan kedalam konstanta K_P, T_i dan T_D melalui tabel 2.1.

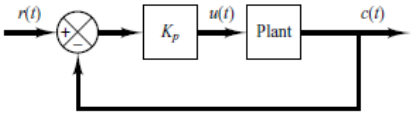
Table 2.1. Aturan *Tuning* metode Ziegler Nichols berdasarkan *Delay Time* dan *Time Constant*

Jenis Kontroler	K_P	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\frac{T}{L}$	$2L$	$0{,}5L$

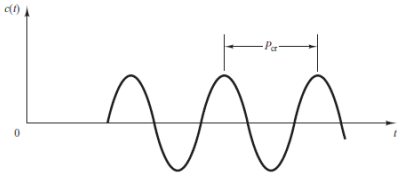
Persamaan *compensator* untuk metode pertama Ziegler Nichols yaitu sebagai berikut:

$$G_c(s) = K_P\left(1 + \frac{1}{T_i s} + T_D s\right)$$
$$G_c(s) = 1.2\,\frac{T_c}{L}\left(1 + \frac{1}{2Ls} + 0.5L\,s\right)$$
$$G_c(s) = 0.6T_c\left(\frac{\left(s+\frac{1}{L}\right)^2}{s}\right) \qquad (2.8)$$

Pada metode kedua, Ziegler dan Nichols mengajukan aturan untuk menentukan nilai-nilai *proportional gain*, *integral time* dan *derivative time* berdasarkan karakteristik respon transien dari sistem. Pertama gain dari T_i dan T_D masing-masing diatur menjadi ∞ dan 0. Selanjutnya nilai Kp dinaikkan dari 0 hingga mencapai nilai kritikal, K_{cr} dimana keluaran pertama kali menunjukkan osilasi berkelanjutan. Nilai gain kritikal K_{cr} dan periode kritikal (P_{cr}) ditentukan secara eksperimen. Berikutnya nilai K_P, T_i dan T_D dapat ditentukan dengan formulasi pada Tabel 2.2.



Gambar 2.16. Sistem Tertutup dengan Pengendali Proporsional



Gambar 2.17. Penentuan Periode Critical (P_{cr})

Tabel 2.2. Aturan *Tuning* metode Ziegler Nichols Berdasarkan parameter *Gain Critical* dan Periode Kritikal

Jenis Kontroler	K_P	T_i	T_d
P	$0.5\ K_{cr}$	∞	0
PI	$0.45\ K_{cr}$	$\frac{1}{1.2}\ P_{cr}$	0
PID	$0.6\ K_{cr}$	$0.5\ P_{cr}$	$0.125\ P_{cr}$

Jika suatu sistem memiliki model matematika yang sudah diketahui fungsi transfer nya , maka metode Root Locus dapat digunakan untuk menentukan nilai dari K_{cr} dan frekuensi dari osilasi, ω_{cr} dimana $P_{cr} = \frac{2\pi}{\omega_{cr}}$. Nilai P_{cr} dapat dicari dari titik persimpangan antara cabang root locus dengan sumbu y pada s-plane seperti pada gambar 2.17. Persamaan *compensator* untuk metode kedua Ziegler-Nichols adalah sebagai berikut:

$$G_c(s) = K_P\left(1 + \frac{1}{T_i s} + T_D s\right)$$
$$G_c(s) = 0.6\ K_{cr}\left(1 + \frac{1}{0.5 P_{cr} s} + 0.125 P_{cr} s\right)$$
$$G_c(s) = 0.075 K_{cr} P_{cr}\left(\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s}\right) \qquad (2.9)$$

2.2.7 Persamaan Gerak Quadcopter

Persamaan gerak quadcopter dibagi menjadi dua yaitu persamaan gerak translasi dan rotasi. Persamaan gerak translasi menghasilkan gerakan translasi quadcopter relatif terhadap *earth frame coordinate*. Parameter keluaran yang dihasilkan persamaan gerak translasi yaitu posisi (x, y, z) beserta derivatifnya. Persamaan

gerak rotasi menghasilkan gerakan rotasi yang relatif terhadap *earth frame coordinate*. Parameter *output* yang dihasilkan persamaan gerak rotasi yaitu posisi sudut $(\emptyset, \theta, \psi)$ beserta derivatifnya.

2.2.7.1 Persamaan Gerak Rotasi Quadcopter

Persamaan gerak rotasi quadcopter mengacu pada Hukum Newton dan Persamaan gerak rotasi Euler. Persamaan gerak rotasi quadcopter pada *Body frame coordinates* diturunkan menjadi:

$$J\dot{\omega} + \omega \times J\omega + M_G = M_B \tag{2.10}$$

Dimana:

- J Inersia polar quadcopter.
- ω perbedaan sudut pada *Body frame coordinates.*
- M_G Momen akibat efek giroskopik.
- M_B Momen yang dihasilkan oleh gaya *thrust* pada motor-propeller.

Pada persamaan 2.10, $(J\dot{\omega} + \omega \times J\omega)$ merupakan kecepatan perubahan momentum angular pada body quadcopter. M_G adalah momen yang dihasilkan oleh perbedaan kecepaan putaran motor sehingga menghasilkan efek momen giroskopik. Bila dijabarkan dengan matrix, besarnya momen giroskopik pada tiap tiap sumbu adalah $\omega \times [0\ 0\ J_r\Omega_r]^T$. Persamaan gerak rotasi quadcopter dapat dijabarkan menjadi:

$$J\dot{\omega} + \omega \times J\omega + \omega \times [0\ 0\ J_r\Omega_r]^T = M_B \tag{2.11}$$

Dimana:

J_r = Inersia motor.

Ω_r = kecepatan relatif keempat motor = $-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$.

2.2.7.2 Inersia Matriks

Inersia pada quadcopter dapat dijabarkan dalam bentuk matriks. Dengan asumsi bentuk quadcopter simetris pada masing-masing sumbu, maka bentuk matriks inersia adalah:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (2.12)$$

I_{xx}, I_{yy}, I_{zz} merupakan momen inersia yang bekerja pada masing-masing sumbu x, y dan z.

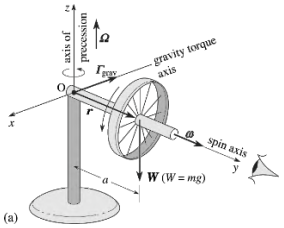
2.2.7.3 Gaya dan Momen yang bekerja pada Quadcopter

- **Efek Gravitasi**

Efek gravitasi merupakan gaya yang timbul karena gravitasi bumi. Gaya gravitasi memiliki arah menuju titik pusat bumi, sejajar dengan sumbu yang tegak lurus dengan permukaan bumi. Formulasinya m.g. g = percepatan gravitasi bumi (9,81 m/s^2).

- **Efek Giroskopik**

Pada setiap benda yang berputar (*spinning*), terjadi satu fenomena unik yang disebut dengan efek giroskopik, dimana pada benda tersebut bekerja gaya dan momen giroskopik. Jika benda tersebut hanya ditinjau sebagai sebuah partikel maka hanya akan ada gaya giroskopik. Adapun jika benda tersebut ditinjau sebagai sebuah rigid body maka disamping gaya giroskopik akan ada pula momen giroskopik. Momen giroskopik merupakan momen yang bekerja akibat ada perbedaan keceptan putaran antara keempat motor pada quadcopter. Adanya perbedaan relatif tersebut menyebabkan adanya momen torsi yang memutar quadcopter pada sumbu z. Efek dari momen giroskopik ditunjukan pada gambar 2.18. Persamaan momen giroskopik merupakan hasil *cross product* antara vector kecepatan angular dengan momentum angular.



Gambar 2.18 Efek Giroskopik

- **Efek Aerodinamis Propeller**

Gaya angkat quadcopter terletak pada gaya *thrust* yang di hasilkan masing-masing motor. Apabila total gaya angkat motor 1-3 berbeda motor 2-4, maka terjadi momen yang menyebabkan quadcopter berotasi pada sumbu y (*pitching).* Apabila total gaya angkat motor 1-2 berbeda dengan motor 3-4 maka terjdi momen yang menyebabkan quadcopter berotasi pada sumbu x (*rolling).* Besarnya gaya angkat dan momen pada masing-masing motor dapat dihitung menggunakan rumus:

$$F_t = C_t \rho \Omega^2 D^4 \quad (2.13)$$

$$M_t = \rho C_p D^5 \Omega^2 \quad (2.14)$$

Dimana:

ρ = massa jenis udara.
D = Diameter propeller.
C_t, C_p = koefisien aerodinamis.
Ω = kecepatan putaran motor.

Pada persamaan 2.13 dan 2.14, beberapa variabel dan konstanta disederhanakan menjadi:

$$F_t = K_f \Omega^2 \quad (2.15)$$

$$M_t = K_M \Omega^2 \quad (2.16)$$

K_f dan K_M merupakan konstanta baru hasil penyederhanaan persamaan 2.13 dan 2.14. Nilai K_f dan K_M dapat ditentukan dengan studi eksperimen. Dari hasil eksperimen John Brandt [5] pada beberapa propeller yang digunakan untuk UAV multirotor berdimensi kecil, didapatkan data koefisien aerodinamis yang bekerja pada propeller APC tipe 1047 seperti pada tabel 2.3. Tabel 2.3 merupakan hubungan konstanta drag dan *thrust* pada propeller APC tipe 1047 pada tingkat putaran tertentu. Peningkatan putaran motor tidak memberikan perubahan yang signifikan pada nilai konstanta drag dan *thrust*. Dari putaran 4029 rpm sampai 6226 rpm

, nilai dari konstanta drag adalah sekitar 0.1 dan konstanta thrust bernilai sekitar 0.05.

Tabel 2.3 Nilai Koefisien Drag dan *Thrust* Propeller APC 1047 [5]

RPM	Ct	Cp
4029	0.1158	0.0466
4319	0.1177	0.0474
4590	0.12	0.0484
4880	0.1223	0.0494
5147	0.1237	0.05
5417	0.1252	0.0508
5715	0.1263	0.0513
5960	0.1278	0.052
6226	0.1286	0.0524

2.2.8 Penurunan Persamaan Gerak Rotasi

Putaran motor-propeller pada quadcopter menyebabkan adanya gaya *thrust*. Masing-masing gaya *thrust* motor akan menghasilkan momen yang merupakan hasil perkalian gaya *thrust* dan jarak titik kerja gaya thrust (F) terhadap titik berat quadcopter. Asumsi titik berat quadcopter berada pada titik pusat koordinat. Jari-jari momen dapat ditentukan sebesar $l_p = l \sin(45)$. Dari gambar 2.19 dapat dijelaskan bahwa total momen yang terjadi pada sumbu x, M_x dapat ditentukan dengan rumus:

$$\begin{aligned} M_x &= (F_3 + F_2)l_p - (F_4 + F_1)l_p \\ M_x &= K_f(\Omega_3^2 + \Omega_2^2)l_p - K_f(\Omega_4^2 + \Omega_1^2)l_p \\ M_x &= l_p K_f\,(\Omega_3^2 + \Omega_2^2 - \Omega_4^2 - \Omega_1^2) \end{aligned} \tag{2.17}$$

Total momen yang bekerja pada sumbu y, M_y dapat ditentukan dengan menggunakan kaidah tangan kanan (*right hand rules)*. Putaran motor 3 dan 4 menghasilkan momen ke arah positif sedangkan putaran motor 1-2 menghasilkan momen ke arah

negatif. Total momen yang dihasilkan ke empat motor terhadap sumbu y adalah:

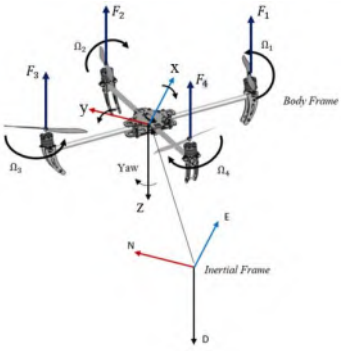
$$\begin{aligned} M_y &= (F_3 + F_4)l_p - (F_2 + F_1)l_p \\ M_y &= K_f(\Omega_3^2 + \Omega_4^2)l_p - K_f(\Omega_2^2 + \Omega_1^2)l_p \\ M_y &= l_p K_f\,(\Omega_3^2 + \Omega_4^2 - \Omega_2^2 - \Omega_1^2) \end{aligned} \tag{2.18}$$

Gaya *thrust* motor tidak berpengaruh pada momen disumbu z, momen pada sumbu z dihasilkan oleh efek momen giroskopik. Persamaa momen terhadap sumbu z dapat dituliskan:

$$\begin{aligned} M_z &= M_1 - M_2 + M_3 - M_4 \\ M_z &= K_M(\Omega_1^2) - K_M(\Omega_2^2) + K_M(\Omega_3^2) - K_M(\Omega_4^2) \\ M_z &= K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{aligned} \tag{2.19}$$

Momen terhadap sumbu x, y dan z dapat ditulis kedalam matrix yaitu:

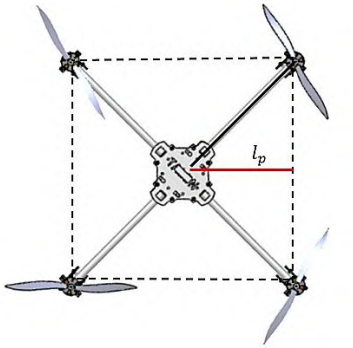
$$M_b = \begin{bmatrix} l_p K_f\,(\Omega_3^2 + \Omega_2^2 - \Omega_4^2 - \Omega_1^2) \\ l_p K_f\,(\Omega_3^2 + \Omega_4^2 - \Omega_2^2 - \Omega_1^2) \\ K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{2.20}$$



Gambar 2.19 Koordinat Quadcopter

l_p, merupakan jarak proyeksi lengan (*arm*) yang tegak lurus terhadap masing-masing sumbu koordinat seperti ilustrasi pada

gambar 2.20. Asumsi bahwa panjang lengan proyeksi quadcopter sama.



Gambar 2.20 Lengan Proyeksi pada Quadcopter

Secara umum, persamaan gerak rotasi quadcopter didapat dengan mensubtitusi persamaan 2.10 dengan 2.12. Persamaan 2.20 disederhanakan menjadi:

$$M_B = \begin{bmatrix} l_p U_2 \\ l_p U_3 \\ U_4 \end{bmatrix} \tag{2.21}$$

Dengan menggunakan inersia matriks, persamaan 2.10 dijabarkan seperti pada persamaan 2.22.

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ J_r \Omega_r \end{bmatrix} = \begin{bmatrix} l_p U_2 \\ l_p U_3 \\ U_4 \end{bmatrix} \tag{2.22}$$

Bila disederhanakan, maka persamaan 2.22 berubah menjadi:

$$\begin{bmatrix} I_{xx}\ddot{\phi} \\ I_{yy}\ddot{\theta} \\ I_{zz}\ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\theta} I_{zz}\dot{\psi} - \dot{\psi} I_{yy}\dot{\theta} \\ \dot{\psi} I_{xx}\dot{\phi} - \dot{\phi} I_{zz}\dot{\phi} \\ \dot{\phi} I_{yy}\dot{\theta} - \dot{\theta} I_{xx}\dot{\phi} \end{bmatrix} + \begin{bmatrix} \dot{\theta} J_r \Omega_r \\ -\dot{\phi} J_r \Omega_r \\ 0 \end{bmatrix} = \begin{bmatrix} l_p U_2 \\ l_p U_3 \\ U_4 \end{bmatrix} \tag{2.23}$$

Dari bentuk matriks persamaan 2.23, didapat masing-masing persamaan gerak rotasi terhadap sumbu putarnya yaitu:

$$\ddot{\phi} = \frac{l_p}{I_{xx}} U_2 - \frac{J_r}{I_{xx}} \dot{\theta}\Omega_r + \frac{I_{yy}}{I_{xx}} \dot{\psi}\dot{\theta} - \frac{I_{zz}}{I_{xx}} \dot{\theta}\dot{\psi} \quad (2.24)$$

$$\ddot{\theta} = \frac{l_p}{I_{yy}} U_3 - \frac{J_r}{I_{yy}} \dot{\phi}\Omega_r + \frac{I_{zz}}{I_{yy}} \dot{\phi}\dot{\psi} - \frac{I_{xx}}{I_{yy}} \dot{\psi}\dot{\phi} \quad (2.25)$$

$$\ddot{\psi} = \frac{1}{I_{zz}} U_4 + \frac{I_{xx}}{I_{zz}} \dot{\theta}\dot{\phi} - \frac{I_{yy}}{I_{zz}} \dot{\phi}\dot{\theta} \quad (2.26)$$

2.2.9 Persamaan Gerak Translasi Quadcopter

Persamaan gerak translasi quadcopter diturunkan berdasarkan Hukum Newton ke-dua pada *inertial frame coordinates*.

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_b \quad (2.27)$$

Dimana:

- r = jarak quadcopter pada koordinat bumi (m).
- m = massa quadcopter (Kg).
- g = percepatan gravitasi bumi (m/s).
- F_b = gaya nongravitasional (N).

Gaya yang bekerja pada quadcopter ketika *level* adalah gaya *thrust* yang bekerja pada masing-masing motor-propeller. Gaya-gaya tersebut menghasilkan percepatan yang melawan percepatan gravitasi. Komponen gaya *thrust* sejajar terhadap sumbu z *body frame coordinates*. persamaan gaya *thrust* dapat dijabarkan kedalam matriks seperti pada persamaan 2.28.

$$F_b = \begin{bmatrix} 0 \\ 0 \\ -K_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \quad (2.28)$$

Komponen gaya pada sumbu x dan y sama dengan nol karena arah gaya *thrust* sejajar dengan sumbu z pada *body frame coordinates*. Tanda negatif pada Fbz menyatakan arah gaya *thrust* tersebut menjauhi pusat bumi dan gaya gravitasi mendekati pusat bumi.

Untuk mendapatkan gerakan translasi pada sumbu x dan y relatif terhadap *inertial frame coordinates*, maka persamaan tersebut dikalikan dengan persamaan rotasi euler. Hasil dari perkalian persamaan gerak translasi dan rotasi euler menghasilkan data gerak absolut terhadap *inertial frame coordinates*.

Persamaan dasar gerak translasi quadcopter didapat dengan mensbutitusikan persamaan (2.27) dan (2.28). persamaan 2.28 dapat disederhanakan menjadi:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} \tag{2.29}$$

Persamaan (2.27) dijabarkan menjadi:

$$m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} c\psi c\theta & c\psi s\phi s\theta & s\phi s\psi + c\phi c\psi s\theta \\ c\theta s\psi & c\theta c\psi + s\phi s\psi s\theta & c\phi s\psi s\theta - c\psi s\theta \\ -s\theta & c\theta s\phi & c\phi c\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix}$$

$$m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} (s\phi s\psi + c\phi c\psi s\theta)(-U_1) \\ (s\phi c\psi - c\phi s\psi s\theta) \\ c\phi c\theta(-U_1) \end{bmatrix} \tag{2.30}$$

Dari Matriks persamaan gerak translasi, didapat persamaan gerak translasi terhadap masing-masing sumbu yaitu:

$$\ddot{x} = \frac{-U_1}{m}(sin\phi\ sin\psi + cos\phi\ cos\varphi\ sin\theta) \tag{2.31}$$

$$\ddot{y} = \frac{-U_1}{m}(cos\phi\ sin\psi\ sin\theta - cos\psi\ sin\phi) \tag{2.32}$$

$$\ddot{z} = g - \frac{U_1}{m}(cos\phi\ cos\theta) \tag{2.33}$$

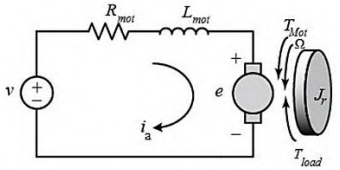
2.2.10 Sistem Dinamis Motor-Propeller

Motor yang digunakan untuk menggerakan quadcopter memiliki tipe brushless V2216-900KV. Motor tersebut dapat menghasilkan torsi besar dan memiliki gaya angkat 500-900 gr.

untuk memodelkan sistem dinamis motor-propeller diperlukan beberapa asumsi antara lain:

- Motor *Brushless* menggunakan *ball bearing* dengan koefisien friksi mendekati nol sehingga gaya gesek antara bearing dengan poros motor dapat diabaikan.
- Motor dalam kondisi *ballance*.
- Pengaruh impedansi lilitan motor dapat diabaikan.

Secara umum, gambar skematik sistem dinamis motor terlihat pada gambar 2.21.



Gambar 2.21 Diagram Sistem Dinamis Motor

Dengan menggunakan Hukum Khircoff, persamaan sistem elektrik motor adalah seperti pada persamaan 2.34 yaitu:

$$v = R_{mot} i_a + L_{mot} \frac{di_a}{dt} + K_{mot} \Omega \tag{2.34}$$

Dimana:

R_{mot} = resistansi kumparan motor.
L_{mot} = Impedansi umparan motor.
i_a = arus listrik yang mengalir pada kumparan.

Dengan menggunakan asumsi yang telah dijelaskan pada paragraf sebelumnya, maka hasil persamaan 2.34 menjadi:

$$v = R_{mot} i_a + K_{mot} \Omega \tag{2.35}$$

K_{mot} adalah konstanta proporsional hubungan antara tegangan *input* dengan putaran *output* tanpa beban (propeller, gesekan, dll). Nilai dari K_{mot} dapat ditentukan dengan menentukan tipe motor

yang digunakan. Motor tipe 2216-900KV memiliki nilai $K_{mot} = \frac{1}{900}$. Nilai K_{mot} dipengaruhi oleh besarnya EMF yang dibangkitkan akibat adanya putaran rotor.

Torsi yang dihasilkan motor dipengaruhi oleh besarnya arus yang mengalir melewati kumparan motor. Besarnya nilai arus motor dapat ditentukan dengan mengubah persamaan 2.35 menjadi:

$$i_a = \frac{v - K_{mot}\Omega_i}{R_{mot}} \tag{2.36}$$

Persamaan gerak motor ditentukan menggunakan *Allemberts law* yaitu:

$$J_r\dot{\Omega}_i = T_{mot} - T_{load} \qquad ; T_{mot} = K_{mot}.\, i_a \tag{2.37}$$

Dengan mensubtitusi persamaan 2.36 kedalam persamaan 2.37, maka didapat persamaan baru yaitu:

$$J_r\dot{\Omega}_i = K_{mot}\left(\frac{v - K_{mot}\Omega_i}{R_{mot}}\right) - K_M\Omega_i^2$$

$$v = \frac{R_{mot}}{K_{mot}}J_r\dot{\Omega}_i + K_{mot}\Omega_i + K_M\Omega_i^2 \tag{2.38}$$

2.2.11 Matriks *Input*

Quadcopter memiliki 4 *input* yaitu U_1 untuk gaya angkat quadcopter, U_2 adalah perbedaan gaya thrust motor 3-2 dengan motor 4-1. U_3 adalah perbedaan thrust motor 3-4 dengan motor 1-2 dan U_4 adalah perbedaan torsi antara motor yang berputar CCW dengan motor yang berputar CW. Matriks *input* dapat dituliskan sebagai berikut:

$$U = [U_1\ U_2\ U_3\ U_4]^T \tag{2.39}$$

Dimana nilai masing-masing U adalah:

$$U_1 = K_f\ (\Omega_3^2 + \Omega_2^2 + \Omega_4^2 + \Omega_1^2)$$
$$U_2 = l.K_f\ (\Omega_3^2 + \Omega_4^2 - \Omega_2^2 - \Omega_1^2)$$
$$U_3 = l.K_f\ (\Omega_3^2 - \Omega_4^2 + \Omega_2^2 - \Omega_1^2)$$
$$U_4 = K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)$$

2.2.12 Representasi State Space pada Persamaan Gerak Quadcopter

Quadcopter merupakan sistem dinamis dengan 4 DOF. State space digunakan untuk memudahkan pemodelan matematis quadcopter serta mengubah nya kedalam *transfer function*. Ada 12 *state variable* yang digunakan untuk memodelkan quadcopter yaitu $(\phi, \theta, \psi, x, y, z)$ beserta derivatif nya, sehingga dapat ditulis menggunakan matrix seperti:

$$[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ \ x_{11}\ x_{12}]^T = [\phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ z\ \dot{z}\ x\ \dot{x}\ y\ \dot{y}]^T \quad (2.40)$$

Quadcopter merupakan sistem dinamis dengan tipe *multi input multi output* (MIMO). Misalnya seperti pada persamaan 2.30, komponen percepatan terhadap sumbu x dipengaruhi oleh beberapa variabel selain *input* momen dari motor, U. Meskipun terdapat *input* momen, namun jika tidak terjadi gerak rotasi pada quadcopter, maka tidak ada percepatan kearah sumbu x. Sehingga, gerakan translasi pada sumbu x dan y pada sistem dinamis quadcopter dapat dikategorikan sebagai *underactuation system* karena pada kenyataanya quadcopter hanya memiliki 4 motor yang akan berpengaruh terhadap 6 persamaan gerak. *Input* yang diberikan keempat motor hanya akan berimbas secara langusng pada gerakan translasi sumbu z dan gerak rotasi terhadap tiap-tiap sumbu koordinat $\{\phi, \theta, \psi\}$. Dua persamaan gerak lainya tergantung oleh orientasi dari quadcopter.

Sistem dinamis quadcopter dimodelkan kedalam state space yang terdiri dari variabel *input*, *state*, dan *output*. Secara umum, state space memiliki perumusan sebagai berikut:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{U}$$
$$\boldsymbol{y} = C\boldsymbol{x} + D\boldsymbol{U} \tag{2.41}$$

Dimana:
x = state vector.
$\dot{x}$ = Derivatif state vector.
y = *output* vector.
U = *input* vector.
A = Matriks sistem.
B = Matriks *input*.
C = Matriks *output*.
D = Matriks *feedforward*.

Dari penurunan persamaan gerak didapat persamaan masing-masing state derivatif pemodelan quadcopter yang terdapat pada persamaan 2.42. Pada persamaan 2.42 dijabarkan 12 state yang mewakili 6 persamaan gerak pada quadcopter yang bergerak relatif terhadap *inertial frame coordinates.* Masing-masing persamaan memiliki variabel trigonometric sehingga persamaan 2.42 merupakan persamaan nonlinear.

$$\begin{pmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \\ \dot{\boldsymbol{x}}_3 \\ \dot{\boldsymbol{x}}_4 \\ \dot{\boldsymbol{x}}_5 \\ \dot{\boldsymbol{x}}_6 \\ \dot{\boldsymbol{x}}_7 \\ \dot{\boldsymbol{x}}_8 \\ \dot{\boldsymbol{x}}_9 \\ \dot{\boldsymbol{x}}_{10} \\ \dot{\boldsymbol{x}}_{11} \\ \dot{\boldsymbol{x}}_{12} \end{pmatrix} = \begin{pmatrix} x_2 \\ \frac{l_p}{I_{xx}} U_2 - \frac{J_r}{I_{xx}} x_4 \Omega_r + \frac{I_{yy}}{I_{xx}} x_4 x_6 - \frac{I_{zz}}{I_{xx}} x_4 x_6 \\ x_4 \\ \frac{l_p}{I_{yy}} U_3 - \frac{J_r}{I_{yy}} x_2 \Omega_r + \frac{I_{zz}}{I_{yy}} x_2 x_6 - \frac{I_{xx}}{I_{yy}} x_2 x_6 \\ x_6 \\ \frac{1}{I_{zz}} U_4 + \frac{I_{xx}}{I_{zz}} x_2 x_4 - \frac{I_{yy}}{I_{zz}} x_2 x_4 \\ x_8 \\ g - \frac{U_1}{m} (cosx_1\ cosx_3) \\ x_{10} \\ \frac{-U_1}{m} (sinx_1\ sinx_5 + cosx_1\ cosx_5\ sinx_3) \\ x_{12} \\ \frac{-U_1}{m} (cosx_1\ sinx_5\ sinx_3 - cosx_5\ sinx_1) \end{pmatrix} \tag{2.42}$$

- **Linearisasi Pemodelan Quadcopter**

Sistem dinamis quadcopter merupakan sistem dinamis dan *non-linear* dengan tipikal MIMO (*Multi Input Multi Output*). Dalam penelitian ini, sistem dinamis quadcopter dibuat linier menggunakan deret taylor dengan rumus:

$$\Delta \dot{x}_i = \sum_{n=1}^{12} \frac{\partial f(X,U)}{\partial x_i} \Delta x_{i,0} + \sum_{m=1}^{4} \frac{\partial f(X,U)}{\partial U_m} \Delta U_{m,0} \qquad (2.43)$$

Linearisasi pemodelan sistem dinamis dilakukan berdasarkan asumsi yaitu:

- Quadcopter bermanuver dengan sudut (*pitch* dan *roll*) $\leq 10^0$
- Simulasi dilakukan pada kondisi *hover* (melayang) dan tanpa adanya gerak translasi maupun rotasi.

Pada kondisi *hover*, masing-masing motor berputar dengan kecepatan yang sama. Perbedaan putaran motor ketika berotasi (*pitch* dan *roll*) cukup kecil sehingga pengaruh efek giroskopik dapat diabaikan, $\Omega_r = 0$. Pemodelan sistem dinamis quadcopter yang telah linier adalah sebagai berikut:

$$\begin{pmatrix} \Delta \dot{x}_1 \\ \Delta \dot{x}_2 \\ \Delta \dot{x}_3 \\ \Delta \dot{x}_4 \\ \Delta \dot{x}_5 \\ \Delta \dot{x}_6 \\ \Delta \dot{x}_7 \\ \Delta \dot{x}_8 \\ \Delta \dot{x}_9 \\ \Delta \dot{x}_{10} \\ \Delta \dot{x}_{11} \\ \Delta \dot{x}_{12} \end{pmatrix} = A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{pmatrix} + B \begin{pmatrix} \Delta U_1 \\ \Delta U_2 \\ \Delta U_3 \\ \Delta U_4 \end{pmatrix} \quad ; y = C \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{pmatrix} + D \begin{pmatrix} \Delta U_1 \\ \Delta U_2 \\ \Delta U_3 \\ \Delta U_4 \end{pmatrix} \qquad (2.44)$$

Dari persamaan 2.44, masing-masing matrix A,B,C dan D adalah:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & l_p/I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & l_p/I_{yy} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/I_{zz} \\ 0 & 0 & 0 & 0 \\ -1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; D = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2.2.15 Analisa Kestabilan

Stabilitas adalah kemampuan sistem mencapai keadaan seimbang saat mendapat *input* atau gangguan. Kestabilan sistem dipengaruhi oleh *natural response* dan *forced response*. *Natural response* adalah cara sistem mendisipasi energi, sedangkan *forced response* adalah respon sistem bergantung pada nilai *input*. Kestabilan sistem dibagi menjadi tiga jenis, sebagai berikut:

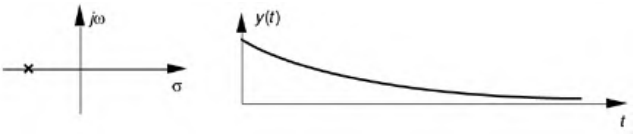
- *Stable system,* terjadi jika respon natural sistem yang mengalami osilasi dapat mencapai nilai nol pada waktu terhingga.
- *Unstable system,* terjadi jika respon natural sistem mengalami kenaikan osilasi terus menerus hingga waktu tak terhingga.

- *Marginally stable*, jika respon natural sistem mengalami osilasi yang tetap hingga mencapai waktu tak terhingga.

Terdapat metode yang bisa digunakan untuk menganalisa kestabilan suatu sistem kontrol, diantaranya adalah dari lokasi pole suatu sistem. Berikut penjelasan respon transien suatu sistem dilihat dari lokasi pole:

- **Negatif real**

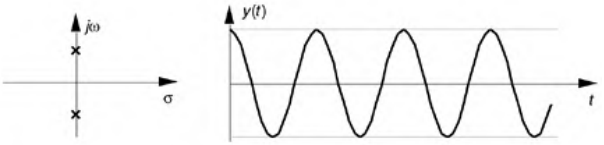
  Sistem dikatakan stabil apabila *poles* berada pada posisi negatif real pada *s-plane*. Pada gambar 2.22 menunjukan lokasi pole berada pada sumbu real negatif.



Gambar 2.22 *Poles* Berada pada Negatif Real

- **Kompleks (real = 0)**

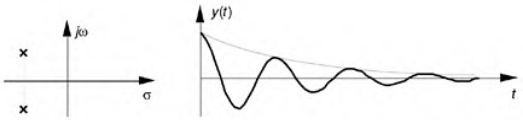
  Jika *pole* berada tepat pada origin *s-plane*, maka respon sistem akan berupa *marginally stable*. Gambar 2.23 menunjukan lokasi pole pada origin berhimpit pada sumbu imajiner.



Gambar 2.23 *Poles* Berada pada *Origin*

- **Negatif real dan kompleks**

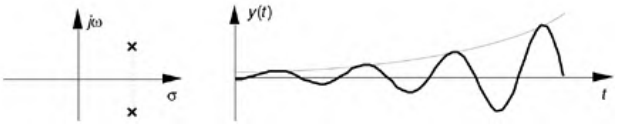
  Jika *pole* bernilai *negative real* dan imajiner, maka respon dari sistem akan berosilasi namun semakin mengecil dan menuju kondisi stabil. Gambar 2.24 menunjukan lokasi poles pada sumbu real negatif dan sumbu imajiner.



Gambar 2.24 *Poles* Bernilai Negatif dan Kompleks

- **Positif**

  Apabila ada satu atau lebih *pole* yang berada pada daerah positif, baik real maupun kompleks, maka sistem cenderung mengalami osilasi yang terus menerus membesar menjadi *unstable system* seperti pada gambar 2.25.



Gambar 2.25 *Pole* Bernilai Positif

(*Halaman ini sengaja dikosongkan)*

BAB 3
METODOLOGI

3.1 Metodologi Tugas Akhir

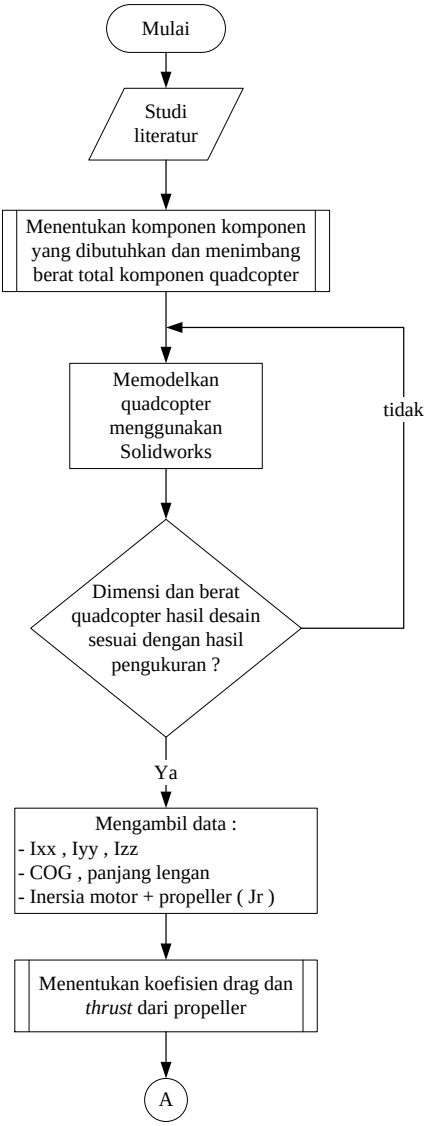
Penelitian ini dimulai dengan mencari data-data yang dibutuhkan untuk menganalisa quadcopter. Data-data yang dibutuhkan untuk proses analisa berupa massa, inersia dan COG (*Center of Gravity*) pada quadcopter. Untuk mendapatkan data-data tersebut, maka bentuk quadcopter dimodelkan kedalam *software* Solidworks. Perbandingan dimensi pemodelan dengan bentuk asli adalah 1:1. Dengan pendekatan tersebut akan didapat nilai dari properti quadcopter seperti massa, inersia, dan COG. Data-data tersebut digunakan sebagai parameter pada pemodelan sistem dinamis quadcopter yang akan disimulasikan kedalam Matlab/simulink.

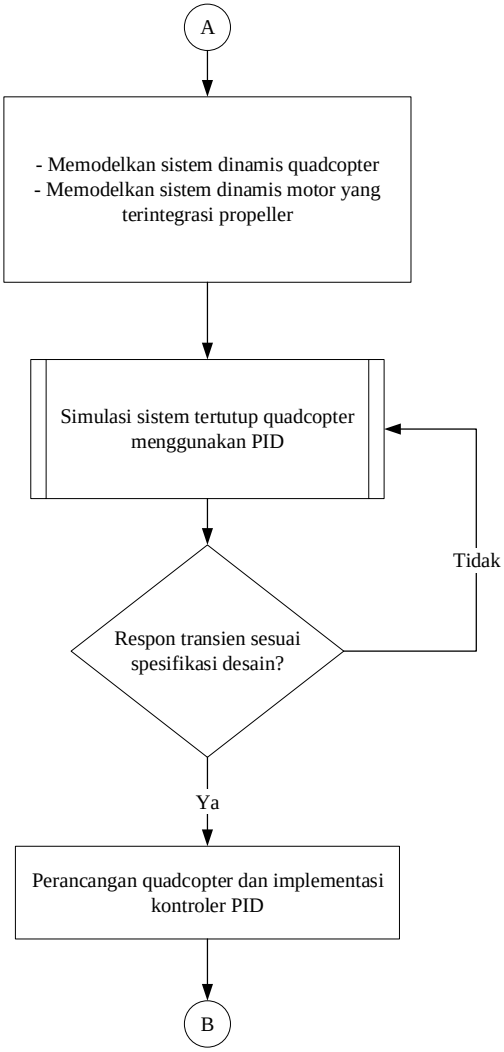
Proses penelitian secara umum dapat dijabarkan sebagai berikut:

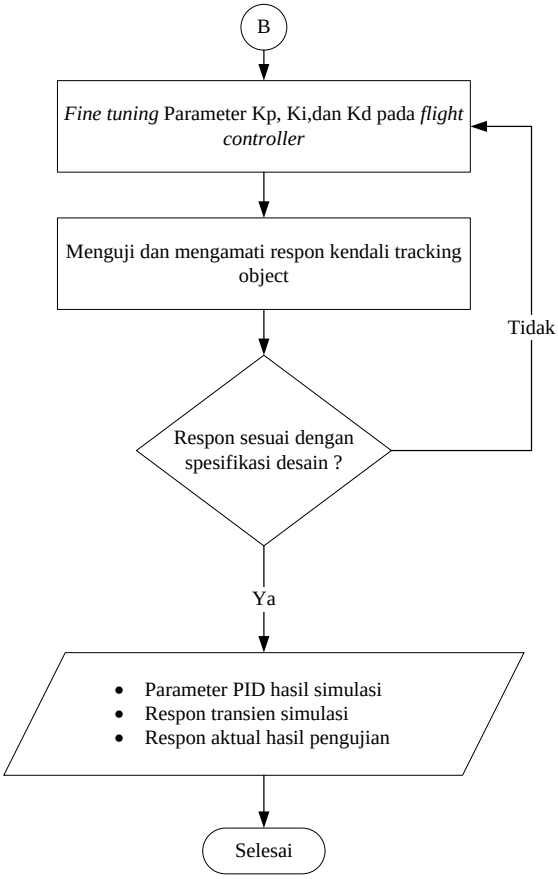
1. Melakukan studi literatur mengenai sistem dinamis quadcopter dan metode kontrol yang sesuai.
2. Memodelkan quadcopter menggunakan software solidworks dengan perbandingan 1:1.
3. Mengambil data-data hasil desain quadcopter yang diperlukan sebagai parameter untuk membuat sistem dinamis quadcopter.
4. Membuat sistem dinamis quadcopter 4 DOF.
5. Memodelkan sistem dinamis motor dan propeller sebagai sistem input pada quadcopter.
6. Menentukan koefisien *drag* dan *thrust* dari propeller berdasarkan hasil eksperimen penelitian terdahulu.
7. Memodelkan sistem dinamis motor-propeller yang telah dilinearisasi.
8. Mensimulasikan sistem secara keseluruhan dan mengamati respon kestabilanya.
9. Menganalisa hasil simulasi dan menyimpulkan.
10. Mengimplementasikan kendali PID kedalam quadcopter.

11. Membuat modul pemancar lokasi.
12. Uji coba sistem kendali tracking object.

3.2 Diagram Alir Tugas Akhir




A
- Memodelkan sistem dinamis quadcopter
- Memodelkan sistem dinamis motor yang terintegrasi propeller
Simulasi sistem tertutup quadcopter menggunakan PID
Respon transien sesuai spesifikasi desain?
Tidak
Ya
Perancangan quadcopter dan implementasi kontroler PID
B

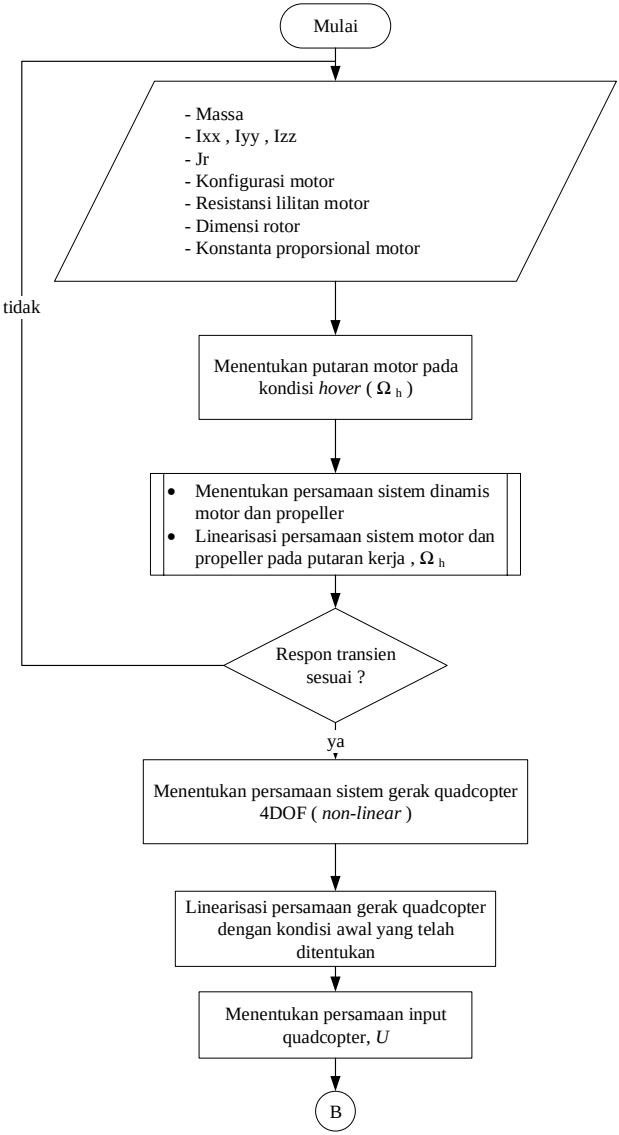

B
Fine tuning Parameter Kp, Ki,dan Kd pada *flight controller*
Menguji dan mengamati respon kendali tracking object
Respon sesuai dengan spesifikasi desain ?
Tidak
Ya
Parameter PID hasil simulasi
Respon transien simulasi
Respon aktual hasil pengujian
Selesai

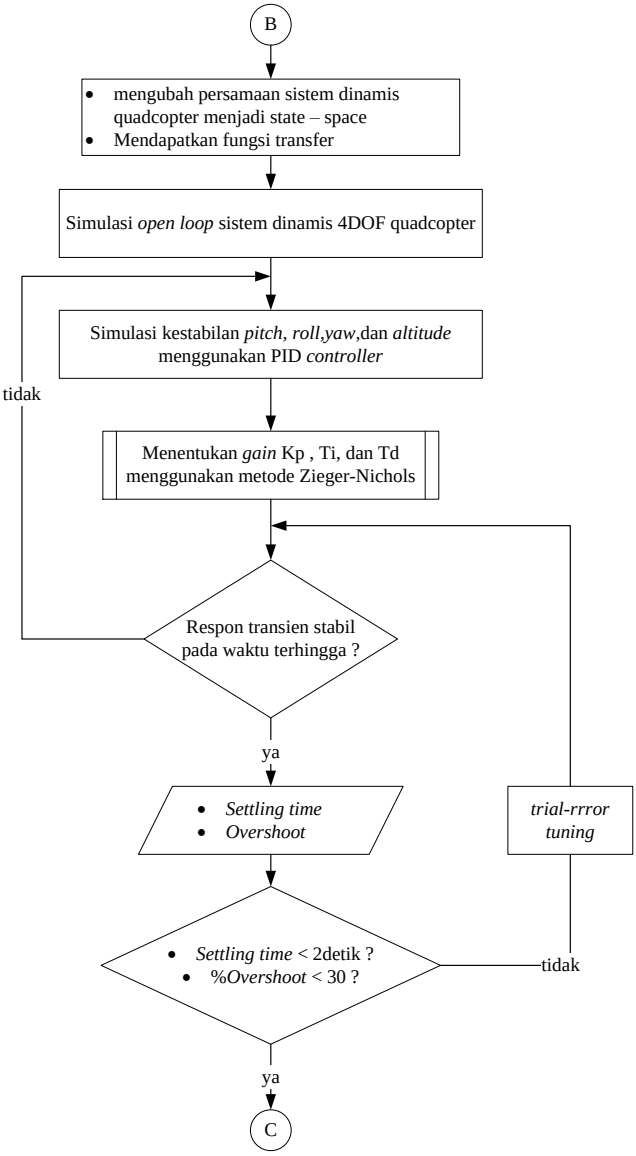
3.3 Metodologi Simulasi Sistem Kendali *Tracking Object*

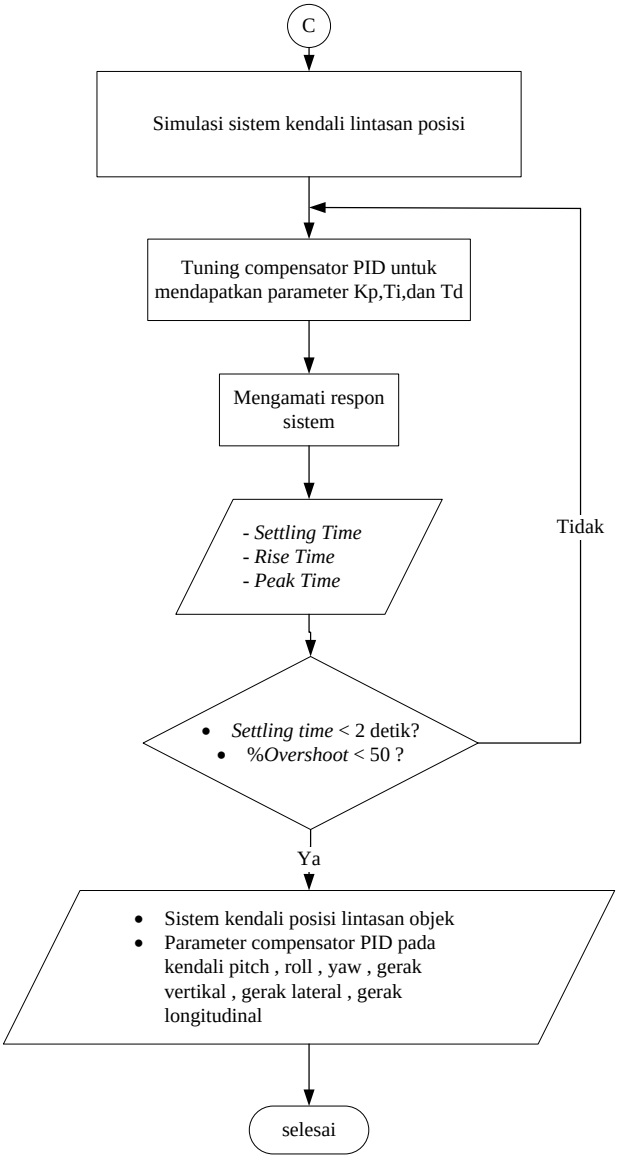
Langkah langkah yang akan dilakukan dalam simulasi dan analisa sistem kendali posisi lintasan quadcopter adalah sebagai berikut:

1. Mencari nilai koefisien *drag* dan *thrust* pada propeller.
2. Menentukan persamaan sistem dinamis motor yang terintegrasi propeller.
3. Linearisasi persamaan sistem dinamis motor-propeller pada kondisi putaran operasional sebesar Ω_{hover}.
4. Menentukan persamaan sistem dinamis quadcopter 4 DOF.
5. Memodelkan persamaan sistem dinamis quadcopter kedalam bentuk state-space.
6. Linearisasi persamaan sistem dinamis quadcopter dengan pendekatan *single input single output*.
7. Simulasi kestabilan *pitch*, *roll*, *yaw*, dan *altitude* menggunakan Matlab Sisotool.
8. Mendapatkan parameter Kp, Ti dan Td yang sesuai.
9. Simulasi sistem kendali lintasan posisi quadcopter dengan *setpoint* jarak 1m.
10. Mendapatkan parameter Kp, Ti dan Td untuk sistem kendali *tracking object*.
11. Mengamati respon transien hasil simulasi.

3.4 Diagram Alir Simulasi




B
mengubah persamaan sistem dinamis quadcopter menjadi state – space
Mendapatkan fungsi transfer
Simulasi *open loop* sistem dinamis 4DOF quadcopter
Simulasi kestabilan *pitch*, *roll*,*yaw*,dan *altitude* menggunakan PID *controller*
tidak
Menentukan *gain* Kp , Ti, dan Td menggunakan metode Zieger-Nichols
Respon transien stabil pada waktu terhingga ?
ya
Settling time
Overshoot
trial-rrror tuning
Settling time < 2detik ?
%*Overshoot* < 30 ?
tidak
ya
C


C
Simulasi sistem kendali lintasan posisi
Tuning compensator PID untuk mendapatkan parameter Kp,Ti,dan Td
Mengamati respon sistem
- Settling Time
- Rise Time
- Peak Time
Tidak
Settling time < 2 detik?
%Overshoot < 50 ?
Ya
Sistem kendali posisi lintasan objek
Parameter compensator PID pada kendali pitch , roll , yaw , gerak vertikal , gerak lateral , gerak longitudinal
selesai

3.5 Kriteria pemilihan komponen

Dalam mengimplementasikan hasil simulasi pada quadcopter, ada beberapa hal yang dapat mempengaruhi performa dan akurasi pengukuran sensor. Beberapa parameter yang perlu diperhatikan dalam memilih komponen quadcopter antara lain:

- ***Magnetic Interference***

Magnetic interference merupakan fenomena adanya medan magnetik yang dihasilkan oleh *power source* seperti baterai dan ESC. *Magnetic interference* berpengaruh pada sensor yang peka terhadap adanya medan magnet seperti sensor *Magnetometer/compass*. Fenomena *magnetic interference* terjadi ketika arus yang dihasilkan baterai berfluktuasi sehingga menimbulkan medan magnet pada kabel. Fluktuasi arus dan tegangan terjadi karena adanya beban motor yang berubah ubah akibat adanya gaya drag dan gaya gesek.

Untuk mengantisipasi pengaruh *magnetic interference* pada *flight controller board*, maka beberapa cara yang harus dilakukan adalah:

a. Menggunakan baterai dengan voltase lebih tinggi.
b. Menggunakan motor sesuai dengan beban (berat) quadcopter.
c. Menjauhkan sensor (*magnetometer*) dan *reciever signal* dari sumber medan magnet
d. Memutar/memelintir kabel penghubung pada baterai.

- **Intensitas Cahaya**

Beberapa sensor tekanan (*barometer*) memiliki kepekaan terhadap intensitas cahaya. Salah satu sensor *barometer* yang sensitif terhadap intensitas cahaya adalah MS5611. Sensor tersebut memiliki kepresisian tinggi dalam mengukur ketinggian, namun sangat peka terhadap paparan sinar matahari.

- **Vibrasi**

Vibrasi merupakan gerakan osilasi quadcopter secara periodik dengan frekuensi dan amplitudo tertentu. Vibrasi berpengaruh pada IMU sensor yang terdiri dari gyro dan accellerometer. Data dari kedua sensor digunakan untuk mengukur sudut absolut quadcopter terhadap bumi. Vibrasi menyebabkan *noise* pada output pengukuran accelerometer dan gyro. Vibrasi dengan amplitudo dan frekuensi yang tinggi menyebabkan quadcopter akan berosilasi tak terkontrol.

- ***Loop time* dan *Refresh Rate* pada MCU**

Implementasi sistem kendali quadcopter terletak pada *flight controller board*. Dalam penulisan program (*sketching*), *flight controller board* mampu menyimpan program dengan kapasitas yang memadai, namun semakin banyak program yang disimpan, maka *loop time* pada *flight controller* akan semakin bertambah. *Loop time* merupakan frekuensi/kecepatan kerja *microcontroller* dalam mengeksekusi dan memproses data dari awal sampai akhir program. *Loop time* berpengaruh pada respon quadcopter. *Loop time* terlalu cepat menyebabkan *noise* pada pengukuran IMU sensor, sedangkan *loop time* terlalu lama menyebabkan respon quadcopter semakin lamban dalam mengkompensasi adanya gangguan maupun adanya perintah.

Refresh rate merupakan kemampuan ESC dalam menerima input/data baru dari *flight controller board*. *Flight controller* mengirim sinyal PWM ke ESC dengan frekuensi/periode tertentu. ESC harus memiliki frekuensi lebih besr dari frekuensi input PWM agar mampu membaca adanya data baru dari *flight controller*.

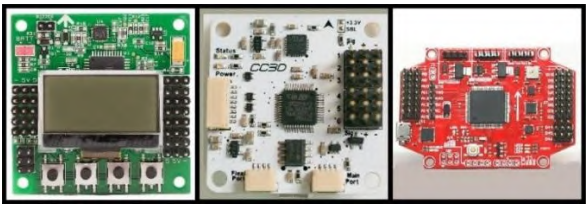
3.6 Komponen Quadcopter

Komponen-komponen quadcopter dipilih berdasarkan parameter-parameter yang telah dijelaskan pada topik sebelumnya. Komponen quadcopter terdiri dari dua bagian yaitu komponen utama dan komponen tambahan.

3.6.1 Komponen Utama

- ***Flight Controller Board***

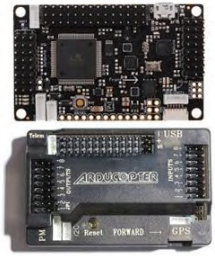
Flight controller board merupakan komponen utama dalam menentukan orientasi dan gerakan quadcopter. *Flight controller board* terdiri dari MCU (*microcontroller unit*) yang diintergasikan dengan beberapa sensor seperti Accelerometer, Gyro, Magnetometer, dan Barometer.



Gambar 3.1 Flight controller board

Pada gambar 3.1, ada beberapa jenis *flight controller* dengan tipe MCU yang berbeda. MCU pada *flight controller board* berperan penting dalam kecepatan pengolahan data dan pemrosesan program yang diinginkan.

Flight controller board yang digunakan pada Tugas Akhir ini adalah Ardupilot Mega (APM 2.6) dengan ilustrasi seperti pada gambar 3.2. Versi board ini terdiri dari sistem minimum Atmega 2560 dengan pin i/o (*input/output*) yang sudah diekspansi untuk keperluan input output data. *Flight controller* ini merupakan *microcontroller* dengan menggunakan *bootloader* milik Arduino tipe Mega. Pada *board* tersebut telah dipasang beberapa sensor sebagai komponen utama dalam mengukur orientasi dan gerakan quadcopter.



Gambar 3.2 APM 2.6

Ardupilot Mega (APM) memiliki spesifikasi sebagai berikut:

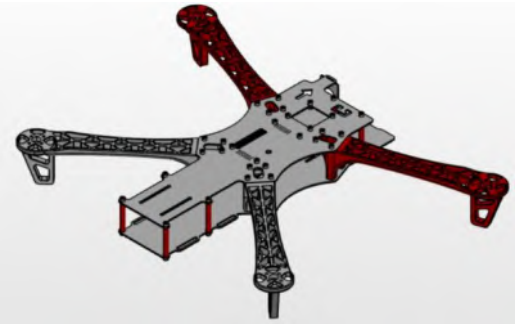
- 8 channel input & output
- IMU (Internal Measurement Unit)
- 3 axis gyro-accelerometer
- Sensor tekanan (barometer)
- *Built-in* Data Logger
- Arduino *programmable*
- *Built-in* relay
- Komunikasi dua arah (dengan modul telemetry)
- Mendukung GPS
- Mendukung Magnetometer
- *Open-Source*

Keunggulan dari APM adalah metode komunikasi serial antara MCU dengan IMU sensor menggunakan SPI (*Serial Peripeheral Interface*). SPI mendukung kecepatan transfer data > 400KHz. Flash memory sebesar 256Kb pada Atmega 2560 membuat APM mampu menyimpan *Sketch* program lebih banyak. Selain itu, APM 2.6 memiliki *enclosure* untuk menutupi bagian sensor yang peka terhadap cahaya seperti sensor barometer (MS5611).

- ***Frame***

Frame merupakan dudukan komponen komponen quadcopter. Pemilihan frame quadcopter didasari oleh optimasi tata letak komponen elektronik seperti motor, ESC, *Power distribution*

Board, baterai, GPS, dan *power module*. Frame yang akan digunakan adalah TBS Discovery Frame (*clone*) seperti pada gambar 3.3.



Gambar 3.3 Frame Quadcopter

Pemilihan frame ini didasarkan oleh tata letak komponen elektronik yang memadahi. *Frame* pada gambar 3.3 memiliki ruang yang sesuai untuk menempatkan komponen-komponen quadcopter serta mampu mereduksi adanya pengaruh *magnetic interference* akibat dari tata letak komponen yang terlalu berdekatan satu sama lain.

- **Kendali Jarak Jauh (*Remote Control*)**

Remote Control Merupakan alat yang digunakan sebagai sistem pengendalian jarak jauh dengan gelombang radio berfrekuensi 2,4 GigaHertz. *Radio Control* ini biasa digunakan untuk aeromodelling (pesawat terbang dan helikopter. Sistem pengendalian jarak jauh dengan menggunakan *Remote Control* terdiri dari 2 unit, pengirim sinyal (*transmitter)* dan penerima sinyal (*receiver)* dan memiliki 9 jalur (*channel*) seperti gambar 3.4.

Transmitter mengirim sinyal PPM (Pulse Position Modulation) menuju *reciever*. *Reciever* mengolah data PPM dan diubah menjadi sinyal PWM. Sinyal PWM dari *reciever* digunakan sebagai input *flight controller*.

Gambar 3.4 Radio Control

- **Electronic Speed Control**

ESC merupakan suatu sirkuit elektrik yang digunakan untuk memvariasikan putaran motor sesuai dengan referensi inputanya. ESC menerima input tegangan PWM dari flight controller dengan rentan 0-5V. Prinsip kerja ESC adalah seperti sirkuit driver elektronik. Variasi inputan dengan rentan 0-5V digunakan untuk memvariasikan tegangan output dengan rentan yang lebih besar (0-12V). Tegangan output dari ESC digunakan untuk memutar motor.

ESC yang digunakan adalah Merk ZTW spider 30A dengan ilustrasi seperti pada gambar 3.5. ESC ini didukung dengan *firmware* Blheli 14.2 yang memiliki kemampuan pembacaan input PWM dengan Frekuensi kerja (*refresh rate*) > 400Hz sehingga mampu merespon perubahan inputan secara cepat.

Gambar 3.5 Elecronic speed Controller ZTW Spider 30A (Blheli *Firmware*)

- **Motor DC Brushless**

Motor DC brushless adalah motor yang dialiri arus searah (*Direct Current/DC)* dan memiliki sistem komutator elektronik, tidak menggunakan komutator mekanik dan sikat (*brushes).*

Hubungan arus-torsi dan frekuensi-kecepatan dari motor DC *brushless* adalah *linear*.

Motor DC yang digunakan adalah merk Sunnysky tipe V2216-900KV. Berikut spesifikasi motor DC brushless Merk sunnysky tipe V2216-900KV dijabarkan seperti pada gambar 3.6.

Specifications	V2216
Stator Diameter	22mm
Stator Thickness	16mm
No. of Stator Arms	12
No. of Stator Poles	14
Motor Kv	900
No-Load Current (A/10V)	0.8A
Motor Resistance	117mΩ
Max Continuous Current	20A/30S
Max Continuous Power	320W
Weight	77g
Outside Diameter	27.7mm
Shaft Diameter	3.175mm
Body Length	34mm
Overall Shaft Length	36.5mm
Max Lipo Cell	2-4S

Gambar 3.6 Spesifikasi Motor Brushless Sunnysky V2216-900KV

- **Propeller**

  Propeller/baling baling merupakan alat untuk mengubah torsi putaran motor menjadi gaya angkat (*Thrust*). Propeller yang digunakan dalam penelitian ini adalah APC 1047 seperti pada gambar 3.7. Propeller ini memiliki tingkat kekakuan yang tinggi yang mampu mengatasi efek *Blade Flapping.*

Gambar 3.7 Propeller APC 1047

3.6.2 Komponen Tambahan

- ***GPS dan Compass***

GPS dan Compass dibutuhkan untuk menentukan koordinat dan orientasi (*Heading*) quadcopter terhadap *inertial frame coordinates* (koordinat Bumi). GPS memberikan koordinat lokasi quadcopter. Data koordinat quadcopter dikirim melalui sebuah *protocol*. Adapun beberapa *protocol* yang tersedia adalah:

- NMEA
- Ublox
- MTK
- SIRF

Informasi yang dapat diterima dari protocol gps adalah *Longitude*, *Lattitude*, *Altitude* (ketinggian), dan waktu (GMT).

Compass (magnetometer) merupakan sensor 3 axis yang digunakan untuk mengukur medan magnet disekitar quadcopter. Data nilai medan magnet masing masing axis digunakan untuk menentukan *Heading* quadcopter relatif terhadap koordinat bumi (*inertial Frame coordinates*). *Compas (magnetometer)* sangat peka terhadap sumber elektromagnet, sehingga diperlukan tata letak komponen yang tepat agar mampu mereduksi adanya gangguan elektromagnet (*magnetic interference*).

Sensor yang akan digunakan pada penelitian ini adalah GPS Ublox M8N + Compass HMC5883L (*built-in*) dengan spesifikasi sebagai berikut:

- Receiver Type:72-channel u-blox M8 engine GPS/QZSS L1 C/A, GLONASS L10F, BeiDou B1
- SBAS L1 C/A: WAAS, EGNOS, MSAS
- Galileo-ready E1B/C (NEO-M8N)
- Nav. Update Rate1 Single GNSS: up to 18 HZ
- Concurrent GNSS: up to 10 Hz
- Position Accuracy:2 2.0 m CEP
- Acquisition2 Cold Starts: 26 s
- Aided Starts: 2 s

- Reacquisition: 1.5 s
- Sensitivity2 Tracking & Nav: –167 dBm
- Cold Starts: –148 dBm
- Hot Starts: –156 dBm
- Operating Temperature: –40° C to 85° C
- Supply Voltage: 1.65 V to 3.6 V (NEO-M8M) 2.7 V to 3.6 V (NEO-M8N/Q)
- Power Consumption:4 23 mA @ 3.0 V (continuous)

Gambar 3.8 GPS M8N dan Compass

- ***Telemetry***

Telemetry merupakan alat/perangkat untuk mentransmisikan data melewati jaringan radio. Perbedaan mendasar *telemetry* dengan *remote control* ada pada metode komunikasi. *Remote control* memiliki sepasang *transmitter* dan *reciever*, sehingga komuniksi yang terjadi adalah satu arah (dari *transmitter* menuju *reciever*). *Telemetry* memiliki dua bagian yaitu *Air Module* dan *Ground Module* seperti yang terlihat pada gambar 3.9. Kedua modul tersebut dapat berfungsi sebagai *transmitter* maupun *reciever*, sehingga komunikasi yang terjadi merupakan koomunikasi dua arah.

Gambar 3.9 Telemetry (*Ground Module* dan *Air Module*)

- ***Power Module***

Power module merupakan perangkat untuk mensuplai tegangan ke *flight controller board*. *Power module* terdiri *switch regulator* dengan tegangan input 5-25V dan tegangan keluaran sebesar 5V. *Power module* mampu memberikan tegangan yang konstan dan dapat mengurangi efek disipasi tegangan menjadi panas akibat perbedaan tegangan input dan output yang besar. *Power module* memiliki efisiensi lebih baik dalam meregulasi tegangan dibandingkan dengan linear regulator. Bila terjadi disipasi tegangan, linear regulator akan memberikan tegangan output lebih rendah dari spesifikasinya. *Power module* yang digunakan merupakan aksesoris dari APM 2.6 dengan ilustrasi seperti pada gambar 3.10.

Gambar 3.10 Power Module dengan Switch Regulator

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan seluruh proses perancangan quadcopter yang terdiri dari beberapa pembahasan meliputi:

- Pemodelan 3D bentuk quadcopter menggunakan software Solidworks.
- Analisa respon transien quadcopter.
- Pemasangan komponen-komponen quadcopter.
- Implementasi sistem kendali quadcopter.

4.1 Pemodelan Quadcopter

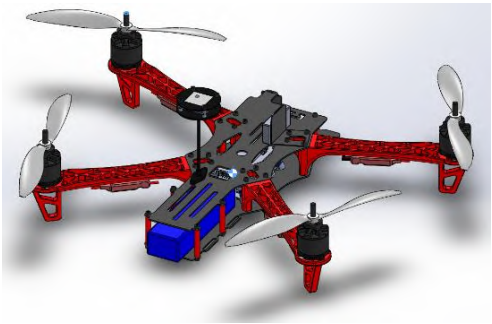
Pemodelan quadcopter dilakukan untuk mendapatkan ilustrasi dan data-data yang diperlukan pada proses simulasi. Dari hasil desain menggunakan solidworks, didapat data-data seperti pada tabel 4.1.

Tabel 4.1 Parameter untuk Pemodelan Sistem Dinamis Quadcopter

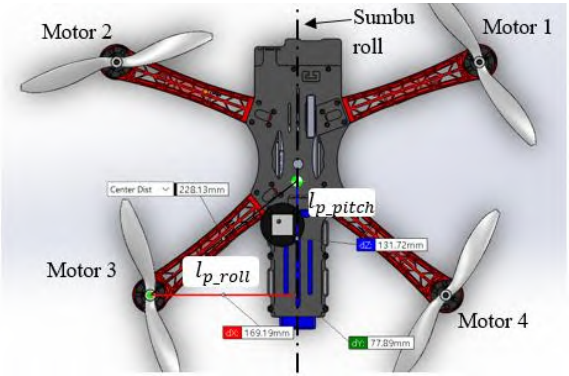
no	Parameter	Unit
1	Massa total quadcopter	1.547 Kg
2	Inersia rotor motor brushless	$4.5878 \times 10^{-5} kg.m^2$
3	Inersia quadcopter	$I_{xx} = 0.023\ kg.m^2$ $I_{yy} = 0.041\ kg.m^2$ $I_{zz} = 0.026\ kg.m^2$
4	Panjang lengan (l)	0.131 m

Gambar 4.1 merupakan hasil desain menggunakan solidworks dengan perbandingan dimensi 1:1. Parameter panjang lengan (l) diukur dari pangkal dimana lengan tersebut dipasang. Bila diproyeksikan kedalam *body frame coordinates*, terdapat dua lengan yang digunakan dalam pemodelan matematis yaitu l_{p_roll} dan l_{p_pitch}. l_{p_roll} merupakan lengan proyeksi yang digunakan untuk pemodelan matematis gerak

roll. l_{p_pitch} merupakan lengan proyeksi yang digunakan untuk pemodelan matematis gerak pitch. Ilustrasi lengan proyeksi ditunjukan pada gambar 4.2.



Gambar 4.1 Hasil Desain Menggunakan Solidworks



Gambar 4.2 Penentuan Lengan Proyeksi gerak Roll dan Pitch

Dari gambar 4.2 terlihat bahwa lengan proyeksi diukur dari titik berat quadcopter. Panjang l_{p_roll} pada motor 3 dan 4 berbeda dengan motor 1 dan 2, namun karena titik berat tepat berada pada sumbu roll, terjadi keseimbangan momen pada gerak roll sehingga putaran masing- masing motor akan sama

yaitu sebesar ω_{hover}. Proses simulasi mengacu pada *single input single output*. Untuk mempermudah analisa, nilai dari l_{p_roll} dapat dianggap sama. Besarnya nilai l_{p_roll} dan l_{p_pitch}berturut turut adalah 169.19 mm dan 131.72 mm.

4.2 Respon Transien pada Sistem Open Loop Quadcopter

Sebelum melakukan simulasi sistem kendali secara *close loop*, diperlukan gambaran sistem secara *open loop* (tanpa sistem kendali). Beberapa proses pemodelan open loop terbagi menjadi dua yaitu:

- Pemodelan sistem input quadcopter.
- Pemodelan sistem dinamis quadcopter.

4.2.1 Pemodelan Sistem Input Quadcopter

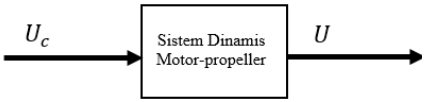
Pada sub bab ini, akan ditentukan hubungan input perintah $U_{c,n}$ terhadap input aktual, U_n dimana n = 1,2,3,4. $U_{c,n}$ merupakan input tegangan total ke empat motor pada sistem dinamis motor-propeller. Dengan pendekatan *single input single output*, sistem yang akan dikendalikan terdiri dari empat subsistem yaitu gerak *rolling*, *pitching*, *yawing* dan gerak vertikal. Masing-masing subsistem berdiri sendiri terhadap sinyal perintah $U_{c,n}$ dan sinyal input aktual U_n. Masing-masing sinyal input perintah, $U_{c,n}$ dijabarkan pada persamaan 4.1.

$$\begin{aligned}
U_{c,1} &= (v_1 + v_2 + v_3 + v_4) \quad ; v_1 = v_2 = v_3 = v_4 \\
U_{c,2} &= (-v_1 + v_2 + v_3 - v_4) \; ; v_1 = v_4 \, , v_2 = v_3 \\
U_{c,3} &= (-v_1 - v_2 + v_3 + v_4) \; ; v_1 = v_3 \, , v_1 = v_4 \\
U_{c,4} &= (v_1 - v_2 + v_3 - v_4) \quad ; v_1 = v_3 \, , v_2 = v_4
\end{aligned} \tag{4.1}$$

Dari persamaan 4.1 terlihat adanya efek *coupling*. Sebagai contoh pada $U_{c,2}$, nilai $v_1 = v_4$ dan $v_2 = v_3$. Artinya putaran motor 1 dan 4 akan sama besar begitu pula pasangan motor 2 dan 3 . Adanya perbedaan putaran masing-masing pasangan motor tersebut menyebabkan momen input U_2 pada sistem

gerak *rolling* quadcopter. Bila $U_{c,2}$ bernilai negatif , artinya pasangan motor 1 dan 4 memiliki putaran motor lebih besar dari pada pasangan motor 2 dan 3. Nilai v_1, v_2, v_3 dan v_4 merupakan masukan tegangan pada masing-masing motor. Batasan masing-masing tegangan masukan motor adalah $0 \leq v \leq v_{bat}$.

Sistem input quadcopter terdiri dari 4 komponen yaitu $\{U_1, U_2, U_3, U_4\}$ dimana U_1 merupakan input gaya angkat (*thrust*) pada subsistem gerak vertikal karena adanya input perintah $U_{c,1}$. U_2 merupakan input momen untuk gerak rotasi *roll* karena adanya input perintah $U_{c,2}$. U_3 adalah input momen yang menghasilkan gerak rotasi *pitch* karena adanya input perintah $U_{c,3}$, dan U_4 merupakan input torsi yang menghasilkan gerak rotasi *yaw* karena adanya input perintah $U_{c,4}$. Hubungan input perintah, U_c dengan U digambarkan pada blok diagram pada gambar 4.3.



Gambar 4.3 Hubungan U_c dengan U

Pada gambar 4.3 terlihat bahwa input perintah, U_c digunakan sebagai input yang masuk kedalam blok fungsi transfer pada sistem dinamis motor-propeller. Keluaran yang dihasilkan oleh blok merupakan input aktual, U yang nantinya masuk kedalam fungsi transfer pada sistem dinamis quadcopter. Gaya angkat , momen, dan torsi merupakan input aktual yang dihasilkan oleh gabungan antara keempat motor pada sistem dinamis motor-propeller. Masing-masing motor berputar dengan beban torsi akibat gaya drag sebesar, $M_t = \rho C_p D^5 \Omega^2$. Persamaan matematis hubungan input tegangan motor dengan putaran motor-propeller terdapat pada persamaan 2.37.Persamaan 2.37 dipindah ruaskan menjadi:

$$\dot{\Omega}_i = [v - K_{mot}.\Omega_i - K_M R_{mot}\Omega_i^2]\frac{K_{mot}}{R_{mot}.J_r} \quad (4.2)$$

Persamaan 2.37 merupakan persamaan *non-linear* karena variabel Ω mengandung fungsi kuadrat. Linearisasi dilakukan untuk mendapatkan persamaan sederhana. Persamaan 4.2 dilinearisasi menggunakan persamaan 2.42 dengan kondisi awal yaitu ketika quadcopter dalam keadaan *hovering*, maka nilai dari $\Omega = \Omega_{hover}$. Dengan menggunakan rumus pada persamaan 2.42, didapat:

$$\Delta\dot{\Omega}_i = [\Delta v - K_{mot}.\Delta\Omega_i - 2.K_M(\Omega_{hover})R_{mot}\Delta\Omega_i]\frac{K_{mot}}{R_{mot}.J_r} \quad (4.3)$$

Besarnya putaran masing-masing motor pada kondisi *hover* didapat dengan persamaan 2.15. Putaran motor pada kondisi hover didapat dengan memenuhi persamaan 4.4.

$$\frac{m.g}{4} = C_t\rho(\Omega_{hover})^2 D^4 \quad (4.4)$$

Besarnya nilai C_t dan Ω_{hover} didapatkan dengan interpolasi data pada tabel 2.4 kemudian disubtitusi kedalam persamaan 4.4. Hasil yang sesuai didapat seperti pada tabel 4.2.

Tabel 4.2 Nilai konstanta pada Sistem Dinamis Motor-propeller

K_f	K_M	R_{mot}	K_{mot}	Ω_{hover}
5.15 $\times 10^{-4}$	5.29 $\times 10^{-5}$	117 mΩ	0.176. 10	86.33 rev/s (542.173 rad/s)

Dengan memasukan nilai masing-masing konstanta pada tabel 4.2, maka persamaan 4.3 menjadi:

$$\Delta\dot{\Omega}_i = \Delta v(1.97 \times 10^3) - \Delta\Omega_i(34.2) \quad (4.5)$$

Fungsi transfer hubungan antara masukan tegangan dengan putaran masing-masing motor-propeller yaitu:

$$\frac{\Delta\Omega_{i(s)}}{\Delta v_{i(s)}} = \frac{1.97 \times 10^3}{s+34.2} \tag{4.6}$$

Persamaan 4.6 didekati dengan model *first order system* menjadi:

$$\frac{\Delta\Omega_{i(s)}}{\Delta v_{i(s)}} = \frac{57.68}{0.03s+1} \tag{4.7}$$

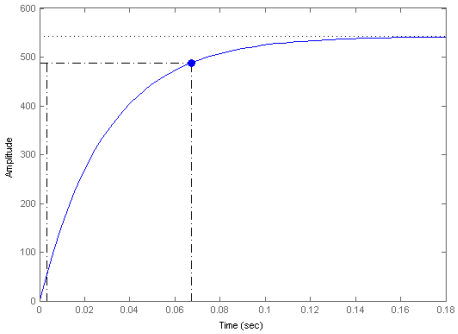
Pada persamaan 4.7 dapat diketahui bahwa *gain* hubungan tegangan dengan putaran motor adalah 57.68. Tegangan motor yang menghasilkan putaran sebesar Ω_{hover} adalah:

$$v_{hover} = \frac{\Omega_{hover}}{gain} \tag{4.8}$$

Persamaan 4.7 disimulasikan kedalam simulink dengan blok diagram seperti pada gambar 4.4. Pada gambar 4.4 terlihat bahwa proses simulasi dimulai dengan memasukan input tegangan berupa step yang diberi *gain* sebesar 9.4. Sehingga total input tegangan yang masuk kedalam fungsi transfer motor adalah sebesar $v_{hover} =$ 9.4 V.

Gambar 4.4 Diagram Simulink pada Pemodelan Motor-propeller

Respon transien hasil simulasi dapat dilihat pada gambar 4.5. Nilai dari *rise time* sebesar 0.067 detik.



Gambar 4.5 Respon Putaran Motor terhadap Input Tegangan Sebesar 9.4V

Momen torsi dan gaya *thrust* yang dihasikan motor didapatkan dengan asumsi:

1. Perubahan koefisien *thrust* dan torsi cukup kecil disekitar putaran *hover*, sehingga dapat dianggap konstan.
2. Hubungan input putaran motor terhadap torsi dianggap berhubungan linier pada daerah disekitar putaran *hover*.

Dengan menggunakan persamaan 4.7, didapat persamaan hubungan input perintah terhadap putaran motor yaitu:

$$\frac{\Delta\Omega}{U_{c,n}} = \frac{57.68}{0.03s+1} \tag{4.9}$$

Output putaran pada persamaan 4.9 diubah menjadi input aktual,U dengan menggunakan persamaan 2.15 dan 2.16 yang telah dilinearisasi pada kondisi kerja (*hover*) yaitu:

$$\begin{aligned} \Delta U_{1;2;3} &= 2.\Omega_{hover}.K_f(\Delta\Omega) \\ \Delta U_4 &= 2.\Omega_{hover}.K_M(\Delta\Omega) \end{aligned} \tag{4.10}$$

Sehingga didapat hubungan input perintah terhadap input aktual yaitu:

$$\frac{\Delta U_1}{U_{c,1}} = \frac{3.2}{0.03s+1} \quad (4.11)$$

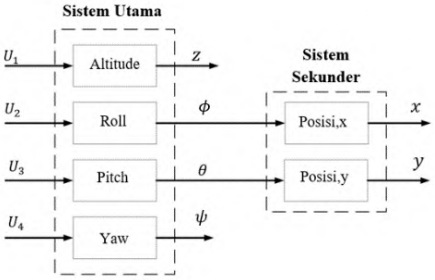
$$\frac{\Delta U_2}{U_{c,2}} = \frac{1.6}{0.03s+1} \quad (4.12)$$

$$\frac{\Delta U_3}{U_{c,3}} = \frac{1.6}{0.03s+1} \quad (4.13)$$

$$\frac{\Delta U_4}{U_{c,4}} = \frac{0.084}{0.03s+1} \quad (4.14)$$

4.2.2 Pemodelan Sistem Dinamis Quadcopter

Pada dasarnya quadcopter memiliki 4 derajat kebebasan yaitu $\{\phi, \theta, \psi, z\}$. Gerakan horizontal quadcopter merupakan akibat adanya gerakan *pitch*, *roll*, dan *yaw*. Berdasarkan hal tersebut, pemodelan sistem dinamis quadcopter dibagi menjadi 2 sistem yaitu sistem utama dan sekunder. Sistem utama merupakan hubungan input *U* terhadap *attitude,heading* dan *altitude*. Subsistem sekunder merupakan hubungan *attitude* terhadap posisi quadcopter.



Gambar 4.6 Dua Bagian pada Sistem Dinamis Quadcopter

Gambar 4.6 merupakan skema pemodelan linear quadcopter dengan pendekatan *single input single output*. Input aktual, U_1 merupakan input gaya *thrust* pada blok fungsi transfer gerak vertikal (*altitude*). Input aktual, U_2 merupakan input momen pada blok fungsi transfer gerak roll. Input aktual, U_3 merupakan input momen pada blok fungsi transfer gerak pitch dan Input aktual, U_4 merupakan input torsi pada blok fungsi

transfer gerak yaw. Masing-masing blok fungsi transfer pada sistem utama didapat dari representasi *state space* pada persamaan 2.43 yang diubah kedalam bentuk fungsi transfer yaitu:

- **Untuk sistem gerak roll**:
  Persamaan gerak:
  $$\ddot{\phi} = \frac{l_p}{I_{xx}} U_2 - \frac{J_r}{I_{xx}} \dot{\theta}\Omega_r + \frac{I_{yy}}{I_{xx}} \dot{\theta}\dot{\psi} - \frac{I_{zz}}{I_{xx}} \dot{\theta}\dot{\psi} \quad (4.15)$$
  Model linear didapatkan dengan menggunakan deret taylor menjadi:
  $$\Delta\ddot{\phi} = \Delta U_2 \frac{l_{p\_roll}}{I_{xx}} \quad (4.16)$$
  Model linear gerak roll diubah dalam bentuk state space menjadi:
  $$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix} \quad (4.17)$$
  $$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{l_p}{I_{xx}} \end{bmatrix} \Delta U_2 \quad (4.18)$$
  $$y_{roll} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (4.19)$$
  Masing-masing matrix pada representasi state space sistem gerak roll adalah:
  $$A_{roll} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}; B_{roll} = \begin{bmatrix} 0 \\ \frac{l_p}{I_{xx}} \end{bmatrix}; C_{roll} = [1 \quad 0]; D_{roll} = 0$$

- **Untuk sistem gerak pitch**:
  Persamaan gerak:
  $$\ddot{\theta} = \frac{l_p}{I_{yy}} U_3 - \frac{J_r}{I_{yy}} \dot{\phi}\Omega_r + \frac{I_{zz}}{I_{yy}} \dot{\phi}\dot{\psi} - \frac{I_{xx}}{I_{yy}} \dot{\phi}\dot{\psi} \quad (4.20)$$
  Model linear didapatkan dengan menggunakan deret taylor menjadi:
  $$\Delta\ddot{\theta} = \Delta U_3 \frac{l_{p\_pitch}}{I_{yy}} \quad (4.21)$$
  Model linear gerak pitch diubah dalam bentuk state space menjadi:

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \quad (4.22)$$

$$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{l_p}{I_{yy}} \end{bmatrix} \Delta U_3 \quad (4.23)$$

$$y_{pitch} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \quad (4.24)$$

Masing-masing matrix pada representasi state space sistem gerak pitch adalah:

$$A_{pitch} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}; B_{pitch} = \begin{bmatrix} 0 \\ \frac{l_p}{I_{yy}} \end{bmatrix}; C_{pitch} = \begin{bmatrix} 1 & 0 \end{bmatrix}; D_{pitch} = 0$$

- **Untuk sistem gerak yaw**:

  Persamaan gerak:

  $$\ddot{\psi} = \frac{1}{I_{zz}} U_4 + \frac{I_{xx}}{I_{zz}} \dot{\phi}\dot{\theta} - \frac{I_{yy}}{I_{zz}} \dot{\phi}\dot{\theta} \quad (4.25)$$

  Model linear didapatkan dengan menggunakan deret taylor menjadi:

  $$\Delta\ddot{\psi} = \frac{1}{I_{zz}} U_4 \quad (4.26)$$

  Model linear gerak yaw diubah dalam bentuk state space menjadi:

  $$\begin{bmatrix} x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} \quad (4.27)$$

  $$\begin{bmatrix} \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{I_{zz}} \end{bmatrix} \Delta U_4 \quad (4.28)$$

  $$y_{yaw} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} \quad (4.29)$$

  Masing-masing matrix pada representasi state space sistem gerak yaw adalah:

  $$A_{yaw} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}; B_{yaw} = \begin{bmatrix} 0 \\ \frac{1}{I_{zz}} \end{bmatrix}; C_{yaw} = \begin{bmatrix} 1 & 0 \end{bmatrix}; D_{yaw} = 0$$

- **Untuk sistem gerak vertikal**:

  Persamaan gerak :

  $$\ddot{z} = g - \frac{U_1}{m}(cos\phi\ cos\theta) \quad (4.30)$$

Model linear didapatkan dengan menggunakan deret taylor menjadi:

$$\Delta\ddot{z} = \Delta U_1 \frac{1}{m} \tag{4.31}$$

Model linear gerak yaw diubah dalam bentuk state space menjadi:

$$\begin{bmatrix} x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \tag{4.32}$$

$$\begin{bmatrix} \dot{x}_7 \\ \dot{x}_8 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \Delta U_1 \tag{4.33}$$

$$y_{alt} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_7 \\ x_8 \end{bmatrix} \tag{4.34}$$

Masing-masing matrix pada representasi state space sistem gerak vertikal adalah:

$$A_{alt} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} ; B_{alt} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} ; C_{alt} = \begin{bmatrix} 1 & 0 \end{bmatrix} ; D_{alt} = 0$$

Matrix A,B,C,D pada masing-masing sistem gerak dimasukan kedalam software MATLAB untuk mendapatkan fungsi transfer. Fungsi transfer yang didapat dari software MATLAB adalah:

$$\frac{\Delta\phi_{(s)}}{\Delta U_{2(s)}} = \frac{5.696}{s^2} \tag{4.35}$$

$$\frac{\Delta\theta_{(s)}}{\Delta U_{3(s)}} = \frac{3.195}{s^2} \tag{4.36}$$

$$\frac{\Delta\psi_{(s)}}{\Delta U_{4(s)}} = \frac{38.46}{s^2} \tag{4.37}$$

$$\frac{\Delta z_{(s)}}{\Delta U_{1(s)}} = \frac{-0.6464}{s^2} \tag{4.38}$$

Tanda negatif pada fungsi transfer persamaan 4.38 adalah karena nilai z positif apabila gerak vertikal berlawanan dengan arah percepatan gravitasi. Dengan menambahkan fungsi transfer motor-propeller, maka persamaan 4.35- 4.38 masing-masing menjadi:

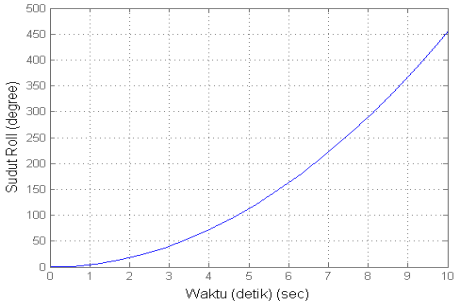
$$\frac{\Delta\phi_{(s)}}{U_{c,2(s)}} = \frac{9.13}{s^2(0.03s+1)} \quad (4.39)$$

$$\frac{\Delta\theta_{(s)}}{U_{c,3(s)}} = \frac{5.12}{s^2(0.03s+1)} \quad (4.40)$$

$$\frac{\Delta\psi_{(s)}}{U_{c,4(s)}} = \frac{3.23}{s^2(0.03s+1)} \quad (4.41)$$

$$\frac{\Delta z_{(s)}}{U_{c,1(s)}} = \frac{-2.048}{s^2(0.03s+1)} \quad (4.42)$$

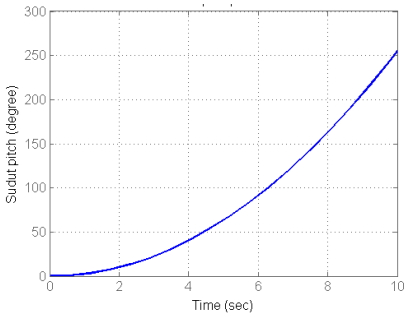
Hasil simulasi masing-masing komponen sistem utama dapat dilihat pada gambar 4.7,4.8,4.9, dan 4.10.



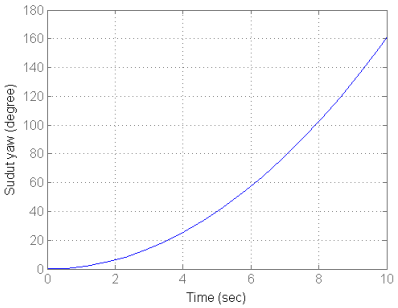
Gambar 4.7 Respon Roll pada Sistem Terbuka

Pada sistem gerak roll, respon yang ditunjukan seperti pada gambar 4.7 dimana grafik yang dihasilkan terus menanjak naik sampai waktu 10 detik. Ketika proses simulasi berjalan 4 detik didapat sudut roll melebihi 10 radian, artinya gerak roll dianggap tidak stabil.Respon gerak pitch ditunjukan oleh gambar 4.8. Pada gambar 4.8 terlihat respon gerak pitch terus bertambah. Pada detik kesepuluh, didapat sudut pitch melebihi 10 radian, artinya sistem gerak pitch tidak stabil.Respon gerak yaw ditunjukan pada gambar 4.9. Gerak rotasi yaw terus meningkat sampai 30 radian pada detik ke lima. Sudut yaw terus bertambah sampai waktu tak hingga. Respon gerak vertikal ditunjukan pada gambar 4.10. Gerak vertikal bernilai positif apabila arah gerakan searah dengan percepatan

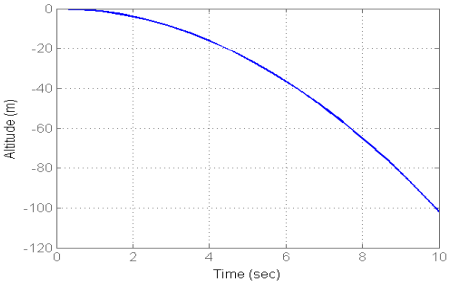
gravitasi sebaliknya, gerak vertikal benilai negatif apabila arah gerakan melawan arah percepatan gravitasi. Grafik menunjukan ketinggian terus menurun sampai detik ke-sepuluh.



Gambar 4.8 Respon Pitch pada Sistem Terbuka



Gambar 4.9 Respon Yaw pada Sistem Terbuka



Gambar 4.10 Respon Altitude pada Sistem Terbuka

4.3 Sistem Kendali pada Sistem Gerak Utama

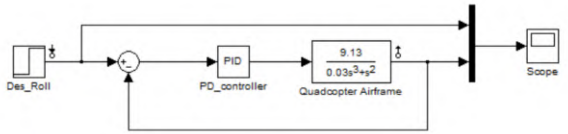
Sistem kendali PID digunakan untuk menstabilkan gerak pada sistem gerak utama yaitu *attitude*, *heading* dan *altitude*. Sinyal keluaran dari PID akan memberikan tegangan tambahan pada masing-masing motor untuk merespon adanya sinyal kesalahan. Kendali PID yang akan digunakan adalah kendali proportional dan derivative. Kendali PD digunakan karena pemodelan sistem dinamis quadcopter tidak memiliki gangguan dari luar (*disturbance*), sehingga kendali Integral tidak digunakan.

Dari data v_{hover}, maka diketahui batasan input tegangan untuk mengkompensasi adanya *error* adalah 6.4V. Sehingga proses tuning harus memperhatikan nilai keluaran yang dihasilkan pada compensator PD. Dalam proses simulasi, respon transien sistem harus mengikuti spesifikasi desain yang ditentukan yaitu:

- *Settling time* < 2 detik.
- *Overshoot* < 30%.

4.3.1 Kendali Roll

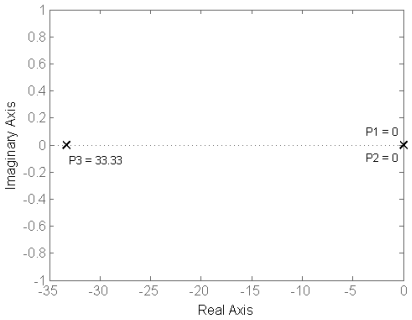
Blok simulasi sistem kendali roll dapat dilihat pada gambar 4.11. Compensator PD dihubungkan pada pemodelan sistem gerak roll. Nilai keluaran pada blok sistem gerak roll akan dibandingkan dengan setpoint untuk mendapatkan nilai *error*. Nilai *error* tersebut akan diolah kembali oleh blok compensator PD.



Gambar 4.11 Blok Simulasi Gerak Roll

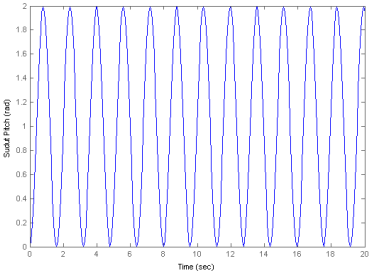
Dari blok persamaan gerak roll, terlihat bahwa sistem gerak roll merupakan tipe *third order system*. Lokasi masing-masing pole terlihat pada gambar 4.12. Pada gambar 4.12 terlihat bahwa terdapat 2 pole yang berhimpit dititik pusat sumbu s-plane yaitu $p_1 = 0$ dan $p_2 = 0$. Terdapat satu pole yang letaknya pada sisi kiri s-plane yaitu $p_3 = -33.34$. Karena letak p_3 sangat jauh

dari pole lainya, p_1, p_2 merupakan pole yang dominan dan pole p_3 dapat diabaikan karena letaknya sangat jauh dari p_1 dan p_2.



Gambar 4.12 Lokasi Pole Sistem Gerak Roll

Proses tuning dilakukan dengan meningatkan nilai K_p sampai terjadi osilasi dengan amplitudo yang konstan pada respon transien gerak roll. Pada gambar 4.13 terlihat bahwa respon gerak roll mengalami osilasi yang konstan dengan amplitudo sebesar 2 radian. Dari hasil simulasi ini didapat parameter-parameter seperti pada tabel 4.3.



Gambar 4.13 Osilasi Gerak Roll

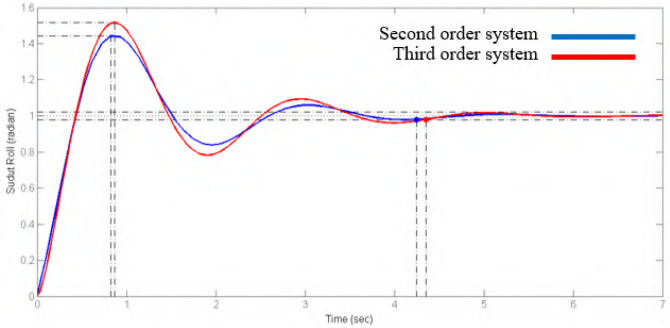
Tabel 4.3 Parameter Metode Ziegler Nichols pada Sistem Gerak Roll

ω_n	P_{cr}	K_{cr}	K_p	T_d
$3.94\ rad/s$	$1.6\ s$	1.7	$0.6\ K_{cr} = 1.02$	$0.125\ P_{cr} = 0.2$

Dari data pada tabel 4.3, maka persamaan compensator PD pada sistem gerak roll adalah sebagai berikut:

$$G_{PD(s)} = K_p(1 + T_d s)$$
$$G_{PD(s)} = 1.02(1 + 0.2\, s) \qquad (4.43)$$

Respon gerak roll yang dihasilkan terlihat pada gambar 4.14. Pada gambar 4.14 terdapat dua grafik respon gerak roll. Grafik warna merah merupakan respon gerak roll pada pemodelan *third order system* sedangan grafik warna biru merupakan respon gerak roll pada pendekatan *second order system*. Bila dibandingkan, respon antara dua grafik tersebut memiliki perbedaan yang kecil sehingga compensator PD dianggap dapat digunakan pada pemodelan *third order system*.



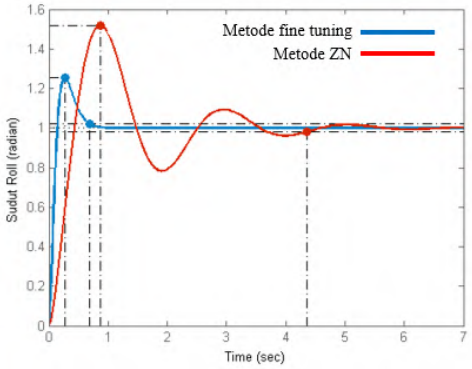
Gambar 4.14 Perbandingan Respon Gerak Roll pada *Third order system* dan *Second Order System*

Dari hasil respon transien pada gambar 4.14, terlihat bahwa sistem gerak roll belum memenuhi spesifikasi desain yang diinginkan. Metode *fine tuning* digunakan untuk memperbaiki respon gerak roll. Proses *fine tuning* dilakukan dengan cara meningkatkan nilai K_p sampai mendapatkan *settling time* sesuai spesifikasi desain. Apabila terjadi *overshoot* yang tinggi, parameter T_d di tingkatkan sampai *overshoot* menurun. Perbandingan parameter K_p dan T_d pada proses tuning terlihat ada tabel 4.4.

Tabel 4.4 Parameter pada Metode Ziegler Nichols dan *Fine Tuning* pada Sistem Gerak Roll

Metode Tuning	K_p	T_d	%OS	*Settling time*
Ziegler-Nichols	1.02	0.2	51.6	4.35
Fine tuning	4.65	0.26	25.6	0.689

Pada tabel 4.4 terlihat bahwa parameter K_p pada metode ziegler-nichols adalah 1.02 sedangkan pada metode *fine tuning* memiliki nilai K_p yang lebih besar yaitu 4.65. Parameter T_d pada metode ziegler nichols adalah 0.2 sedangkan pada metode *fine tuning* parameter T_d meningkat yaitu sebesar 0.26. *Overshoot* yang dihasilkan metode ziegler nichols yaitu sebesar 51.6 persen sedangkan pada proses fine tuning presentase *overshoot* menurun menjadi 25.6 persen. *Settling time* yang dihasilkan metode ziegler nichols sebesar 4.35 detik sedangkan metode *fine tuning* menghasilkan *settling time* yang singkat yaitu sebesar 0.689 detik. Perbandingan grafik respon transien kedua metode ditunjukan pada gambar 4.15. Grafik warna merah merupakan respon gerak roll dengan metode tuning ziegler-nichols sedangkan grafik warna biru merupakan respon gerak roll dengan metode fine tuning.



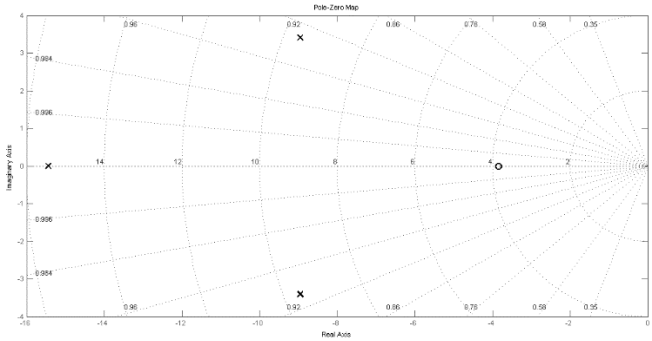
Gambar 4.15 Respon Gerak Roll dengan Metode Tuning Ziegler-Nichols dan Fine Tuning

Persamaan gerak roll pada sistem terbuka yang telah diberi compensator PD terlihat pada persamaan 4.44.

$$\frac{\emptyset_{(s)}}{U_{2(s)}} = \frac{11.4s + 42.47}{0.03s^3 + s^2} \tag{4.44}$$

Lokasi pole ditunjukan pada gambar 4.16. Gambar 4.16 merupakan lokasi pole pemodelan sistem tertutup gerak roll. Sistem gerak roll merupakan sistem yang stabil karena lokasi pole berada pada sisi kiri s-plane. Masing-masing lokasi pole dan zero adalah sebagai berikut:

- Pole:
  -15.4368 ; -8.9483 + 3.4062i ; -8.9483 - 3.4062i
- Zero:
  -3.8462

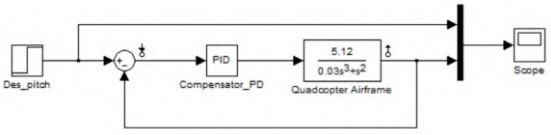


Gambar 4.16 Lokasi Pole dan Zero Sistem Tertutup Gerak Roll dengan Compensator PD

4.3.2 Kendali Pitch

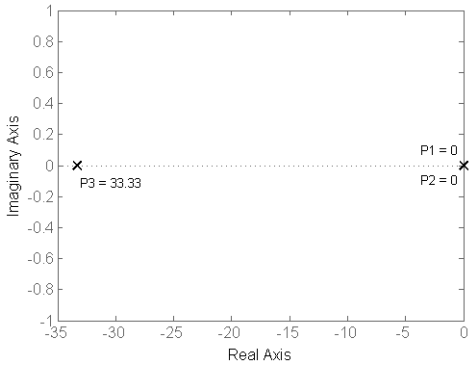
Blok simulasi sistem kendali pitch dapat dilihat pada gambar 4.17. Compensator PD dihubungkan pada pemodelan sistem gerak pitch. Nilai keluaran pada blok sistem gerak pitch akan dibandingkan dengan setpoint untuk mendapatkan nilai *error*.

Nilai *error* tersebut akan diolah kembali oleh blok compensator PD.



Gambar 4.17 Blok Simulasi Gerak Pitch

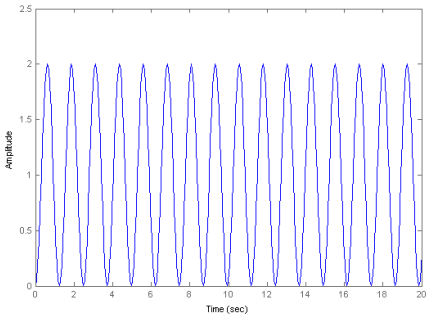
Dari blok persamaan gerak pitch, terlihat bahwa sistem gerak pitch merupakan tipe *third order system*. Lokasi masing-masing pole terlihat pada gambar 4.18. Pada gambar 4.18 terlihat bahwa terdapat 2 pole yang berhimpit dititik pusat sumbu s-plane yaitu $p_1 = 0$ dan $p_2 = 0$. Terdapat satu pole yang letaknya pada sisi kiri s-plane yaitu $p_3 = -33.34$. Karena letak p_3 sangat jauh dari pole lainya, p_1, p_2 merupakan pole yang dominan dan pole p_3 dapat diabaikan karena letaknya sangat jauh dari p_1 dan p_2.



Gambar 4.18 Lokasi Pole Sistem Gerak Pitch

Proses tuning dilakukan dengan meningatkan nilai K_p sampai terjadi osilasi dengan amplitudo yang konstan pada respon transien gerak pitch. Pada gambar 4.19 terlihat bahwa respon gerak pitch mengalami osilasi yang konstan dengan amplitudo sebesar 2

radian. Dari hasil simulasi ini didapat parameter-parameter seperti pada tabel 4.5.



Gambar 4.19 Osilasi Gerak Pitch

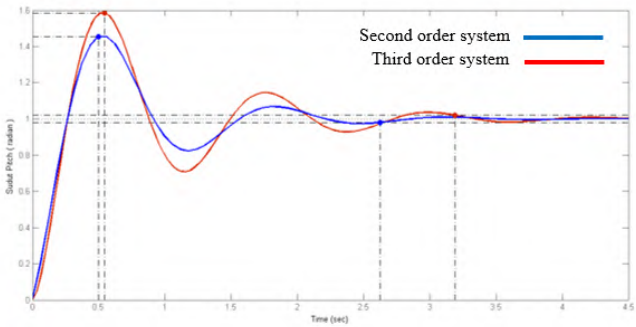
Tabel 4.5 Nilai Parameter Metode Ziegler Nichols pada Sistem Gerak Pitch

ω_n	P_{cr}	K_{cr}	K_p	T_d
$5.06\ rad/s$	$1.24\ s$	5	$0.6\ K_{cr} = 3$	$0.125\ P_{cr} = 0.115$

Dari data pada tabel 4.3, maka persamaan compensator PD pada sistem gerak roll adalah:

$$G_{PD(s)} = K_p(1 + T_d s)$$
$$G_{PD(s)} = 3(1 + 0.115\ s) \quad (4.45)$$

Respon gerak pitch yang dihasilkan terlihat pada gambar 4.20. Pada gambar 4.20 terdapat dua grafik respon gerak pitch. Grafik warna merah merupakan respon gerak pitch pada pemodelan *third order system* sedangan grafik warna biru merupakan respon gerak pitch pada pendekatan *second order system*. Bila dibandingkan, respon antara dua grafik tersebut memiliki perbedaan yang kecil sehingga compensator PD dianggap dapat digunakan pada pemodelan *third order system*.



Gambar 4.20 Perbandingan Respon Gerak Pitch pada *Third order system* dan *Second Order System*.

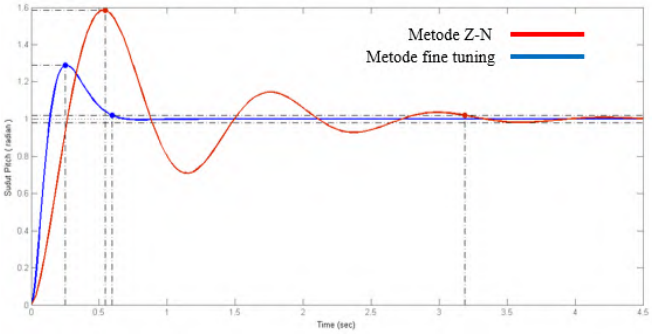
Dari hasil respon transien pada gambar 4.20, terlihat bahwa sistem gerak pitch belum memenuhi spesifikasi desain yang diinginkan. Metode fine tuning digunakan untuk memperbaiki respon gerak pitch. Proses fine tuning dilakukan dengan cara meningkatkan nilai K_p sampai mendapatkan *settling time* sesuai spesifikasi desain. Apabila terjadi *overshoot* yang tinggi, parameter T_d di tingkatkan sampai *overshoot* menurun. Perbandingan parameter K_p dan T_d pada proses tuning terlihat ada tabel 4.4.

Tabel 4.6 Parameter pada Metode Ziegler Nichols dan Fine Tuning pada Gerak Pitch

Metode Tuning	K_p	T_d	%OS	*Settling time*
Ziegler-Nichols	3	0.115	58.5	3.19
Fine tuning	10	0.22	29	0.565

Pada tabel 4.6 terlihat bahwa parameter K_p pada metode ziegler-nichols adalah 3 sedangkan pada metode fine tuning memiliki nilai K_p yang lebih besar yaitu 10. Parameter T_d pada metode ziegler nichols adalah 0.115 sedangkan pada metode fine tuning parameter T_d meningkat yaitu sebesar 0.22. *Overshoot* yang dihasilkan

metode ziegler nichols yaitu sebesar 58.5 persen sedangkan pada proses fine tuning presentase *overshoot* menurun menjadi 29 persen. *Settling time* yang dihasilkan metode ziegler nichols sebesar 3.19 detik sedangkan metode fine tuning menghasilkan *settling time* yang singkat yaitu sebesar 0.565 detik. Perbandingan grafik respon transien kedua metode ditunjukan pada gambar 4.21. Grafik warna merah merupakan respon gerak pitch dengan metode tuning ziegler-nichols sedangkan grafik warna biru merupakan respon gerak pitch dengan metode fine tuning.



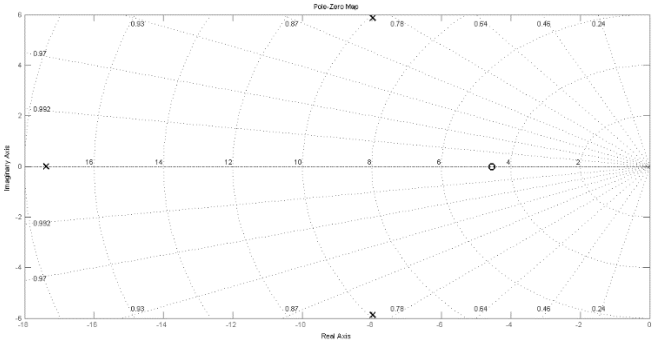
Gambar 4.21 Respon Gerak Pitch dengan Metode Tuning Ziegler-Nichols dan Fine Tuning

Persamaan gerak pitch pada sistem terbuka yang telah diberi compensator PD terlihat pada persamaan 4.46.

$$\frac{\theta_{(s)}}{U_{3(s)}} = \frac{11.26s+51.2}{0.03s^3+s^2} \tag{4.46}$$

Lokasi pole ditunjukan pada gambar 4.22. Gambar 4.22 merupakan lokasi pole pada pemodelan sistem tertutup gerak pitch. Sistem gerak pitch merupakan sistem yang stabil karena lokasi pole berada pada sisi kiri s-plane. Masing-masing lokasi pole dan zero adalah sebagai berikut:

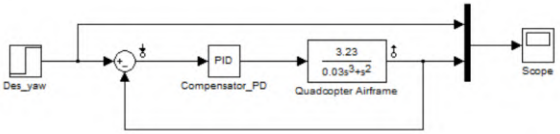
- Pole:
  -17.3799; -7.9767 + 5.8796i; -7.9767 - 5.8796i
- Zero:
  -4.5455



Gambar 4.22 Lokasi Pole dan Zero Sistem Tertutup Gerak Pitch dengan Compensator PD

4.3.3 Kendali Yaw/*Heading*

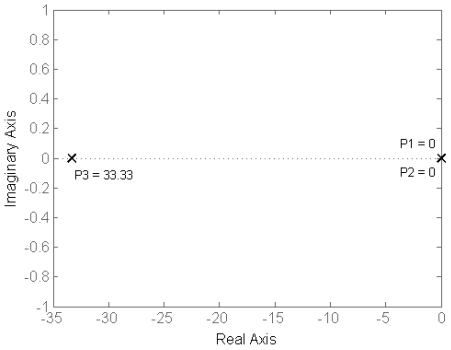
Blok simulasi sistem kendali yaw dapat dilihat pada gambar 4.23. Compensator PD dihubungkan pada pemodelan sistem gerak yaw. Nilai keluaran pada blok sistem gerak yaw akan dibandingkan dengan setpoint untuk mendapatkan nilai *error*. Nilai *error* tersebut akan diolah kembali oleh blok compensator PD.



Gambar 4.23 Blok Simulasi Gerak Yaw

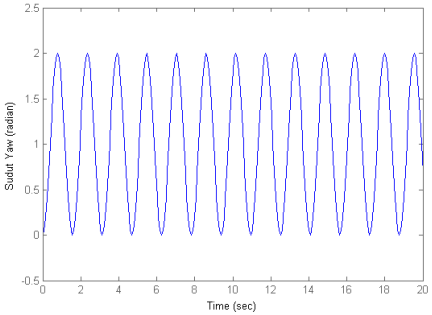
Dari blok persamaan gerak yaw, terlihat bahwa sistem gerak yaw merupakan tipe *third order system*. Lokasi masing-masing pole terlihat pada gambar 4.24. Pada gambar 4.24 terlihat bahwa terdapat 2 pole yang berhimpit dititik pusat sumbu s-plane

yaitu $p_1 = 0$ dan $p_2 = 0$. Terdapat satu pole yang letaknya pada sisi kiri s-plane yaitu $p_3 = -33.34$. Karena letak p_3 sangat jauh dari pole lainya, p_1, p_2 merupakan pole yang dominan dan pole p_3 dapat diabaikan karena letaknya sangat jauh dari p_1 dan p_2.



Gambar 4.24 Lokasi Pole Sistem Gerak Yaw

Proses tuning dilakukan dengan meningatkan nilai K_p sampai terjadi osilasi dengan amplitudo yang konstan pada respon transien gerak yaw. Pada gambar 4.25 terlihat bahwa respon gerak yaw mengalami osilasi yang konstan dengan amplitudo sebesar 2 radian. Dari hasil simulasi ini didapat parameter-parameter seperti pada tabel 4.7.



Gambar 4.25 Osilasi Gerak Yaw

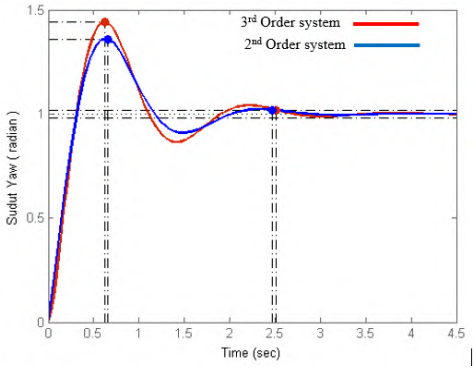
Tabel 4.7 Nilai Parameter Metode Ziegler Nichols pada Sistem Gerak Yaw

ω_n	P_{cr}	K_{cr}	K_p	T_d
$4.02\ rad/s$	$1.56\ s$	5	$0.6\ K_{cr} = 3$	$0.125\ P_{cr} = 0.195$

Dari data pada tabel 4.7, maka persamaan compensator PD pada sistem gerak yaw adalah:

$$G_{PD(s)} = K_p(1 + T_d s)$$
$$G_{PD(s)} = 3(1 + 0.195\ s) \qquad (4.47)$$

Respon gerak yaw yang dihasilkan terlihat pada gambar 4.26. Pada gambar 4.26 terdapat dua grafik respon gerak yaw. Grafik warna merah merupakan respon gerak yaw pada pemodelan *third order system* sedangan grafik warna biru merupakan respon gerak yaw pada pendekatan *second order system*. Bila dibandingkan, respon antara dua grafik tersebut memiliki perbedaan yang kecil sehingga compensator PD dianggap dapat digunakan pada pemodelan *third order system*.



Gambar 4.26 Perbandingan Respon Gerak Yaw pada *Third order system* dan *Second Order System*.

Dari hasil respon transien pada gambar 4.26, terlihat bahwa sistem gerak yaw belum memenuhi spesifikasi desain yang diinginkan.

Metode *fine tuning* digunakan untuk memperbaiki respon gerak yaw. Proses *fine tuning* dilakukan dengan cara meningkatkan nilai K_p sampai mendapatkan *settling time* sesuai spesifikasi desain. Apabila terjadi *overshoot* yang tinggi, parameter T_d di tingkatkan sampai *overshoot* menurun. Perbandingan parameter K_p dan T_d pada proses tuning terlihat ada tabel 4.8.

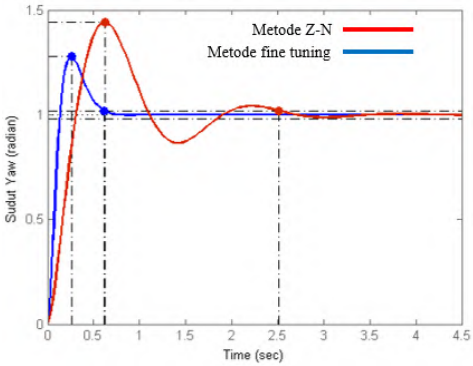
Tabel 4.8 Parameter Metode Ziegler Nichols dan *Fine tuning* pada Sistem Gerak Yaw

Metode Tuning	K_p	T_d	%OS	*Settling time*
Ziegler-Nichols	3	0.195	44	2.51
Fine tuning	15.6	0.23	27.5	0.611

Pada tabel 4.8 terlihat bahwa parameter K_p pada metode ziegler-nichols adalah 3 sedangkan pada metode *fine tuning* memiliki nilai K_p yang lebih besar yaitu 15.6. Parameter T_d pada metode ziegler nichols adalah 0.195 sedangkan pada metode *fine tuning* parameter T_d meningkat yaitu sebesar 0.23. *Overshoot* yang dihasilkan metode ziegler nichols yaitu sebesar 44 persen sedangkan pada proses *fine tuning* presentase *overshoot* menurun menjadi 27.5 persen. *Settling time* yang dihasilkan metode ziegler nichols sebesar 2.51 detik sedangkan metode *fine tuning* menghasilkan *settling time* yang singkat yaitu sebesar 0.611 detik. Perbandingan grafik respon transien kedua metode ditunjukan pada gambar 4.27. Grafik warna merah merupakan respon gerak yaw dengan metode tuning ziegler-nichols sedangkan grafik warna biru merupakan respon gerak yaw dengan metode *fine tuning*.

Persamaan gerak yaw pada sistem terbuka yang telah diberi compensator PD terlihat pada persamaan 4.48.

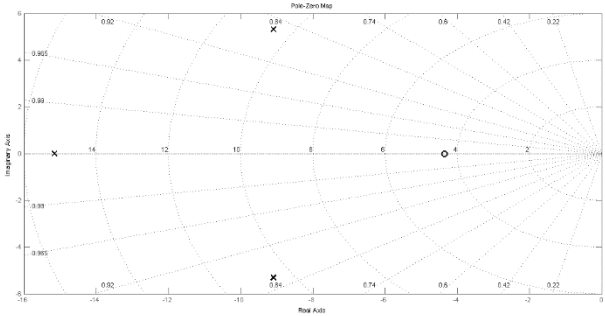
$$\frac{\varphi_{(s)}}{U_{3(s)}} = \frac{11.59s+50.39}{0.03s^3+s^2} \tag{4.48}$$



Gambar 4.27 Respon Gerak Yaw dengan Metode Tuning Ziegler-Nichols dan *Fine tuning*

Lokasi pole ditunjukan pada gambar 4.28. Gambar 4.28 merupakan lokasi pole pada pemodelan sistem tertutup gerak yaw. Sistem gerak yaw merupakan sistem yang stabil karena lokasi pole berada pada sisi kiri s-plane. Masing-masing lokasi pole dan zero adalah sebagai berikut:

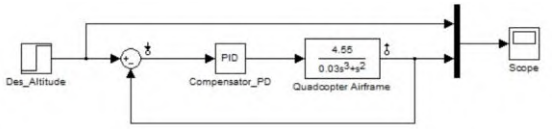
- Pole:
  -15.1531; -9.0901 + 5.3119i; -9.0901 - 5.3119i
- Zero:
  -4.3477



Gambar 4.28 Lokasi Pole dan Zero Sistem Tertutup Gerak Yaw dengan Compensator PD.

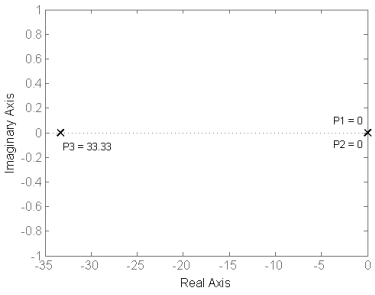
4.3.4 Kendali Gerak Vertikal

Blok simulasi sistem kendali ketinggian (gerak vertikal) dapat dilihat pada gambar 4.29. Compensator PD dihubungkan pada pemodelan sistem gerak vertikal. Nilai keluaran pada blok sistem gerak vertikal akan dibandingkan dengan setpoint untuk mendapatkan nilai *error*. Nilai *error* tersebut akan diolah kembali oleh blok compensator PD.



Gambar 4.29 Blok Simulasi Gerak Vertikal

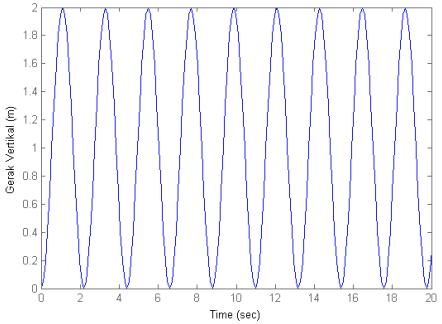
Dari blok persamaan gerak vertikal, terlihat bahwa sistem gerak vertikal merupakan tipe *third order system*. Lokasi masing-masing pole terlihat pada gambar 4.30. Pada gambar 4.30 terlihat bahwa terdapat 2 pole yang berhimpit dititik pusat sumbu s-plane yaitu $p_1 = 0$ dan $p_2 = 0$. Terdapat satu pole yang letaknya pada sisi kiri s-plane yaitu $p_3 = -33.34$. Karena letak p_3 sangat jauh dari pole lainya, p_1, p_2 merupakan pole yang dominan dan pole p_3 dapat diabaikan karena letaknya sangat jauh dari p_1 dan p_2.



Gambar 4.30 Lokasi Pole pada Sistem Gerak vertikal

Proses tuning dilakukan dengan meningatkan nilai K_p sampai terjadi osilasi dengan amplitudo yang konstan pada respon transien

gerak vertikal. Pada gambar 4.31 terlihat bahwa respon gerak vertikal mengalami osilasi yang konstan dengan amplitudo sebesar 2 radian. Dari hasil simulasi ini didapat parameter-parameter seperti pada tabel 4.9.



Gambar 4.31 Osilasi Gerak Vertikal

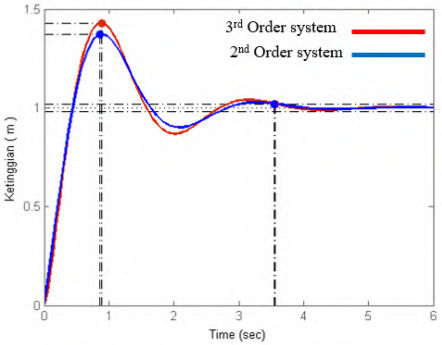
Tabel 4.9 Parameter Metode Ziegler Nichols pada Sistem Gerak Vertikal

ω_n	P_{cr}	K_{cr}	K_p	T_d
$2.86\ rad/s$	$2.195\ s$	4	$0.6\ K_{cr} = 2.4$	$0.125\ P_{cr} = 0.27$

Dari data pada tabel 4.9, maka persamaan compensator PD pada sistem gerak vertikal adalah:

$$G_{PD(s)} = K_p(1 + T_d s)$$
$$G_{PD(s)} = 2.4(1 + 0.27\ s) \qquad (4.49)$$

Respon gerak vertikal yang dihasilkan terlihat pada gambar 4.32. Pada gambar 4.32 terdapat dua grafik respon gerak vertikal. Grafik warna merah merupakan respon gerak vertikal pada pemodelan *third order system* sedangan grafik warna biru merupakan respon gerak vertikal pada pendekatan *second order system*. Bila dibandingkan, respon antara dua grafik tersebut memiliki perbedaan yang kecil sehingga compensator PD dianggap dapat digunakan pada pemodelan *third order system*.



Gambar 4.32 Perbandingan Respon Gerak Vertikal pada *Third order system* dan *Second Order System*.

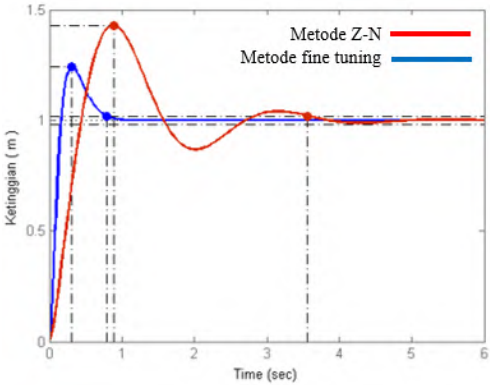
Dari hasil respon transien pada gambar 4.32, terlihat bahwa sistem gerak vertikal belum memenuhi spesifikasi desain yang diinginkan. Metode *fine tuning* digunakan untuk memperbaiki respon gerak vertikal. Proses *fine tuning* dilakukan dengan cara meningkatkan nilai K_p sampai mendapatkan *settling time* sesuai spesifikasi desain. Apabila terjadi *overshoot* yang tinggi, parameter T_d di tingkatkan sampai *overshoot* menurun. Perbandingan parameter K_p dan T_d pada proses tuning terlihat ada tabel 4.10.

Tabel 4.10 Perbandingan Parameter pada Metode Ziegler Nichols dengan *Fine tuning* pada Sistem Gerak Vertikal

Metode Tuning	K_p	T_d	%OS	*Settling time*
Ziegler-Nichols	4	0.27	42.9	3.56
Fine tuning	16.42	0.29	24.3	0.78

Pada tabel 4.10 terlihat bahwa parameter K_p pada metode ziegler-nichols adalah 4 sedangkan pada metode *fine tuning* memiliki nilai K_p yang lebih besar yaitu 16.42. Parameter T_d pada metode ziegler nichols adalah 0.27 sedangkan pada metode *fine tuning* parameter T_d meningkat yaitu sebesar 0.29. *Overshoot* yang dihasilkan

metode ziegler nichols yaitu sebesar 42.9 persen sedangkan pada proses *fine tuning* presentase *overshoot* menurun menjadi 24.3 persen. *Settling time* yang dihasilkan metode ziegler nichols sebesar 3.56 detik sedangkan metode *fine tuning* menghasilkan *settling time* yang singkat yaitu sebesar 0.78 detik. Perbandingan grafik respon transien kedua metode ditunjukan pada gambar 4.33. Grafik warna merah merupakan respon gerak vertikal dengan metode tuning ziegler-nichols sedangkan grafik warna biru merupakan respon gerak vertikal dengan metode *fine tuning*.



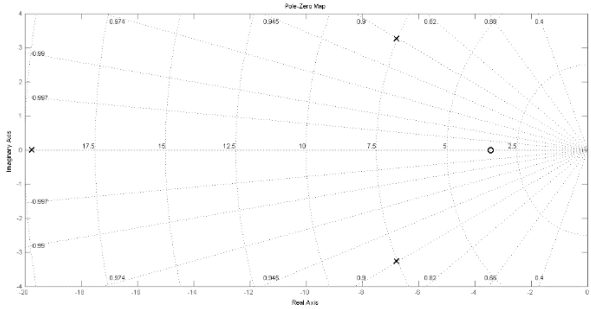
Gambar 4.33 Respon Gerak Vertikal dengan Metode Tuning Ziegler-Nichols dan *Fine tuning*

Persamaan gerak vertikal pada sistem terbuka yang telah diberi compensator PD terlihat pada persamaan 4.50.

$$\frac{Z_{(s)}}{U_{1(s)}} = \frac{9.752s+0.59}{0.03s^3+s^2} \tag{4.50}$$

Lokasi pole ditunjukan pada gambar 4.34. Gambar 4.34 merupakan lokasi pole pada pemodelan sistem tertutup gerak vertikal. Sistem gerak vertikal merupakan sistem yang stabil karena lokasi pole berada pada sisi kiri s-plane. Masing-masing lokasi pole dan zero adalah sebagai berikut:

- Pole:
  -19.7450; -6.7942 + 3.2573i; -6.7942 - 3.2573i
- Zero:
  -3.4483



Gambar 4.34 Lokasi Pole dan Zero Sistem Tertutup Gerak Vertikal dengan Compensator PD

4.3.5 Kendali Posisi

Simulasi kendali posisi dibagi menjadi dua yaitu:

- Kendali posisi longitudinal (horizontal-x): gerak translasi quadcopter pada sumbu-x.
- Kendali posisi lateral (horizontal-y): gerak translasi quadcopter pada sumbu-y.

Kendali posisi longitudinal berhubungan dengan gerak pitch sedangkan kendali lateral berhubungan dengan gerakan roll. Spesifikasi desain yang dibutuhkan adalah *overshoot* < 50% pada setpoint 1m dan *settling time* < 2 detik. Persamaan gerak longitudinal dan lateral dapat ditulis sebagai berikut:

- Persamaan gerak longitudinal:
  $\ddot{X} = \frac{-U_1}{m}(sin\phi\ sin\psi + cos\phi\ cos\psi\ sin\theta)$ (4.51)
  Model linear didapatkan dengan menggunakan deret taylor menjadi:
  $\Delta\ddot{X} = g.\Delta\theta$ (4.52)

Persamaan 4.32 diubah menjadi fungsi transfer yaitu:

$$\frac{\Delta X_{(s)}}{\Delta \theta_{(s)}} = \frac{g}{s^2} \quad (4.53)$$

- Persamaan gerak lateral:

$$\ddot{Y} = \frac{-U_1}{m}(cos\phi\ sin\psi\ sin\theta - cos\psi\ sin\phi) \quad (4.54)$$

Model linear didapatkan dengan menggunakan deret taylor menjadi:

$$\Delta\ddot{Y} = g.\Delta\phi \quad (4.55)$$

Persamaan 4.35 diubah menjadi fungsi transfer yaitu:

$$\frac{\Delta Y_{(s)}}{\Delta \phi_{(s)}} = \frac{g}{s^2} \quad (4.56)$$

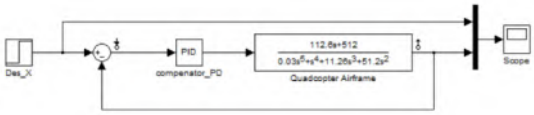
Pada persamaan 4.53 dan 4.56 terlihat bahwa model linear gerak longitudinal dan lateral merupakan efek dari gerak pitch dan roll. Dengan menggabungkan sistem gerak pitch dan roll pada persamaan 4.53 dan 4.56, maka hubungan input aktual terhadap gerak longitudinal dan lateral adalah:

$$\frac{\Delta Y_{(s)}}{\Delta U_{2(s)}} = \frac{114s+424.7}{0.03s^5+ s^4+11.4s^3+42.47s^2} \quad (4.57)$$

$$\frac{\Delta X_{(s)}}{\Delta U_{3(s)}} = \frac{112.6s+512}{0.03s^5+ s^4+11.26s^3+51.2s^2} \quad (4.58)$$

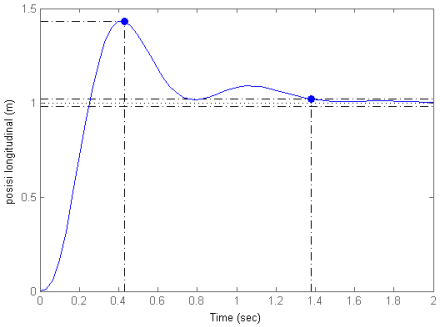
4.3.5.1 Kendali Posisi Longitudinal

Blok simulasi sistem kendali posisi longitudinal dapat dilihat pada gambar 4.35. Compensator PD dihubungkan pada pemodelan sistem gerak longitudinal. Nilai keluaran pada blok sistem gerak longitudinal akan dibandingkan dengan setpoint untuk mendapatkan nilai *error*. Nilai *error* tersebut akan diolah kembali oleh blok compensator PD.



Gambar 4.35 Blok Simulasi Gerak Longitudinal

Proses trial *error* tuning dilakukan untuk mendapatkan respon sesuai desain spesifikasi. Parameter K_p ditentukan terlebih dahulu, bila sistem tidak stabil maka parameter T_d ditingkatkan sampai sistem stabil. Hasil tuning ditunjukan pada gambar 4.36.



Gambar 4.36 Respon Gerak Longitudinal dengan Compensator PD

Pada gambar 4.36 terlihat bahwa respon transien gerak longitudinal pada quadcopter mengalami *overshoot* sebesar 40 persen dan *settling time* pada 1.38 detik. Persamaan compensator PD pada gerak longitudinal terdapat pada persamaan 4.59.

$$G_{PD(s)} = K_p(1 + T_d s)$$
$$G_{PD(s)} = 1.191(1 + 0.43\ s) \qquad (4.59)$$

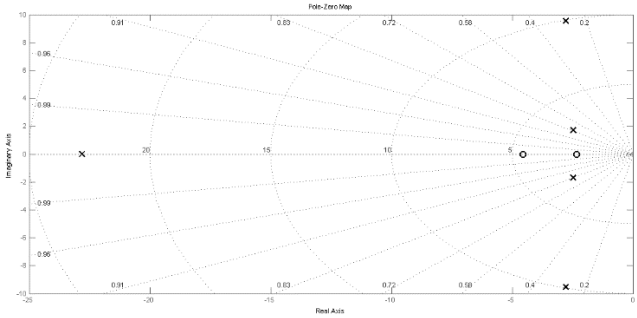
Sehingga, persamaan fungsi transfer sistem terbuka pada gerak longitudinal terdapat pada persamaan 4.60.

$$\frac{X_{(s)}}{U_{3(s)}} = \frac{57.67s^2+396.3s+609.8}{0.03s^5+ s^4+11.26s^3+51.2s^2} \qquad (4.60)$$

Lokasi pole ditunjukan pada gambar 4.37. Gambar 4.37 merupakan lokasi pole pada pemodelan sistem tertutup gerak longitudinal. Sistem gerak longitudinal merupakan sistem yang stabil karena

lokasi pole berada pada sisi kiri s-plane. Masing-masing lokasi pole dan zero adalah sebagai berikut:

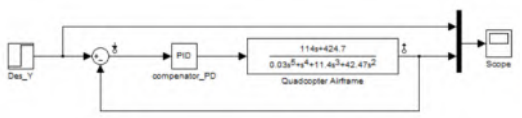
- Pole:
  -22.8168; -2.7795 + 9.5435i; -2.7795 - 9.5435i; -2.4787 + 1.6948i; -2.4787 - 1.6948i
- Zero:
  -4.5471; -2.3256



Gambar 4.37 Lokasi Pole dan Zero Sistem Tertutup Gerak Longitudinal dengan Compensator PD

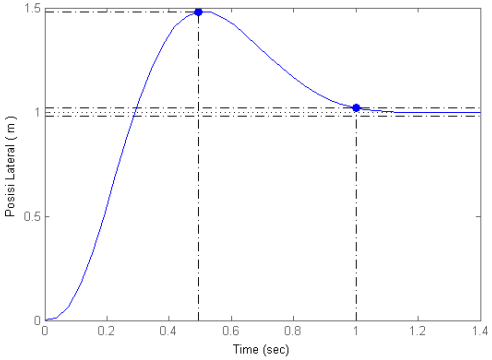
4.3.5.2 Kendali Posisi Lateral

Blok simulasi sistem kendali posisi lateral dapat dilihat pada gambar 4.38. Compensator PD dihubungkan pada pemodelan sistem gerak lateral. Nilai keluaran pada blok sistem gerak lateral akan dibandingkan dengan setpoint untuk mendapatkan nilai *error*. Nilai *error* tersebut akan diolah kembali oleh blok compensator PD



Gambar 4.38 Blok Simulasi Gerak Lateral

Proses trial *error* tuning dilakukan untuk mendapatkan respon sesuai desain spesifikasi. Parameter K_p ditentukan terlebih dahulu, bila sistem tidak stabil maka parameter T_d ditingkatkan sampai sistem stabil. Hasil tuning ditunjukan pada gambar 4.39.



Gambar 4.39 Respon Gerak Lateral dengan Compensator PD

Pada gambar 4.39 terlihat bahwa respon transien gerak lateral pada quadcopter mengalami *overshoot* sebesar 48 persen dan *settling time* pada 1 detik. Persamaan compensator PD pada gerak lateral terdapat pada persamaan 4.61.

$$G_{PD(s)} = K_p(1 + T_d s)$$
$$G_{PD(s)} = 1.241(1 + 0.3\, s) \qquad (4.61)$$

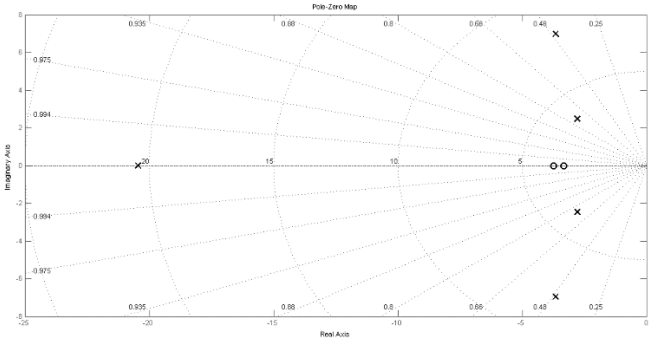
Sehingga, persamaan fungsi transfer sistem terbuka pada gerak lateral terdapat pada persamaan 4.62.

$$\frac{Y_{(s)}}{U_{3(s)}} = \frac{42.44s^2 + 299.6s + 527.1}{0.03s^5 + s^4 + 11.4s^3 + 42.47s^2} \qquad (4.62)$$

Lokasi pole ditunjukan pada gambar 4.40. Gambar 4.40 merupakan lokasi pole pada pemodelan sistem tertutup gerak longitudinal. Sistem gerak longitudinal merupakan sistem yang stabil karena

lokasi pole berada pada sisi kiri s-plane. Masing-masing lokasi pole dan zero adalah sebagai berikut:

- Pole:
  -20.4537; -3.6591 + 6.9731i; -3.6591 - 6.9731i; -2.7807 + 2.4735i; -2.7807 - 2.4735i
- Zero:
  -3.7254; -3.3333

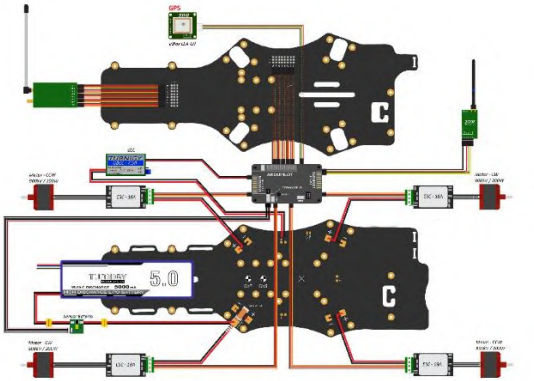


Gambar 4.40 Lokasi Pole dan Zero Sistem Tertutup Gerak Lateral dengan Compensator PD.

4.4 Proses Perancangan Quadcopter

Proses perancangan quadcopter dimulai dengan merangkai komponen – komponen utama maupun pendukung seperti yang dipaparkan pada bab 3. Skema konfigurasi komponen-komponen quadcopter terlihat pada gambar 4.41.

Hasil perancangan terlihat pada gambar 4.42. Pada gambar 4.43 terlihat adanya action camera yang digunakan untuk mendokumentasikan lingkungan sekitar ketika quadcopter berada diudara. Visual gambar yang dihasilkan oleh action camera ditransmisikan ke layar monitor pada ground station dengan menggunakan video transmitter.

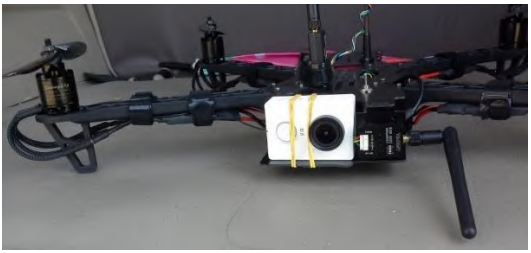


Gambar 4.41 Rangkaian Komponen pada Quadcopter

4.5 Proses Perancangan Sistem Kendali pada Quadcopter

Proses perancangan sistem kendali quadcopter dimulai dari konfigurasi parameter PID pada modul APM2.6. Parameter proportional ditingkatkan sampai respon gerak quadcopter berosilasi. Parameter derivative ditingkatkan sampai osilasi berkurang. Untuk meningkatkan respon gerak, parameter proportional kembali ditingkatkan sampai respon gerak meningkat. Parameter integral ditingkatkan dengan tinggkat penambahan 0.01 pada setiap kali penambahan. Proses tuning PID dilakukan menggunakan software mission planner seperti pada gambar 4.44.

Gambar 4.42 Hasil Perancangan Quadcopter dan Modul Pemancar Lokasi



Gambar 4.43 Penambahan Action Camera pada Quadcopter

Gambar 4.44 Antar Muka Software Mission Planner

Modul APM 2.6 memiliki beberapa mode terbang yang dapat dimanfaatkan dalam mengambangkan sistem autopilot pada UAV Antara lain:

- Loiter: menahan posisi
- RTL: proses autopilot untuk kembali ke lokasi awal quadcopter dinyalakan
- Guide: bergerak menuju satu titik lokasi yang telah ditentukan

Perancangan kendali *tracking object* dilakukan dengan memanfaatkan mode terbang yang ada pada APM 2.6 yaitu mode

"Guide". Proses perubahan mode terbang dapat dilakukan menggunakan AUX *switch* yang ada pada remote control. Mode terbang yang akan digunakan adalah Loiter dan Guide. Loiter digunakan untuk inisialisasi kondisi awal. Mode Guide digunakan untuk kendali *tracking object*. Proses penentuan lokasi dilakukan oleh module pemancar lokasi. Module pemancar lokasi akan mengirimkan lokasi target setiap 2 detik sekali. Quadcopter akan bergerak menuju lokasi yang telah ditentukan. Apabila quadcopter telah tiba dilokasi sebelum 2 detik, maka quadcopter akan menahan posisi sampai mendapat data lokasi baru.

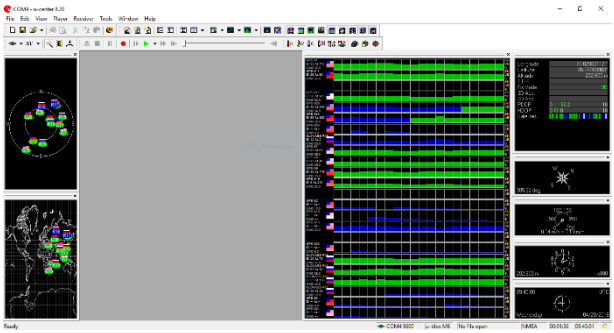
4.6 Perancangan Modul Pemancar Lokasi

Modul pemancar lokasi digunakan untuk mengirim data posisi global suatu target yang akan diterima oleh *flight controller*. Proses pemancaran lokasi dilakukan dengan jeda beberapa detik. Proses perancangan modul pemancar lokasi dibagi menjadi beberapa proses. Setelah menentukan komponen-komponen elektronik, maka proses perancangan modul adalah sebagai berikut:

- **Konfigurasi GPS eksternal**

Untuk dapat digunakan sesuai kebutuhan, perlu dilakukan konfigurasi terhadap modul gps eksternal. Konfigurasi gps dilakukan menggunakan software U-center. Konfigurasi dilakukan dengan mengatur baudrate sesuai dengan kebutuhan. Baudrate yang tidak sinkron menyebabkan terjadi kegagalan komunikasi terhadap GPS. Antar muka software U-center terlihat pada gambar 4.45.

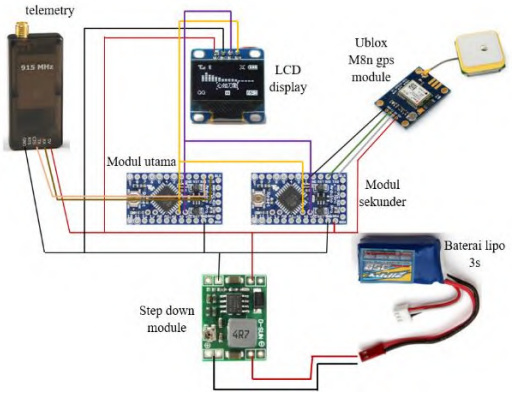
Pengukuran performance gps dapat dilihat pada informasi satelit yang terkunci oleh perangkat GPS. Semakin banyak satelit yang terkunci, maka tingkat akurasi data lokasi akan semakin tinggi.Rata rata akurasi gps module ketika jumlah satelit yang terkunci lebih dari 10 adalah radius $\pm 90\ cm$.



Gambar 4.45 Antar Muka Software U-center

4.7 Integrasi Komponen pada Module Pemancar Lokasi

Dalam merakit modul pemancar lokasi , dibutuhkan dua modul arduino. Modul arduino yang digunakan adalah arduino promini 5V dengan clock 16Mhz. Modul pertama digunakan sebagai modul sekunder. Modul sekunder digunakan untuk berkomunikasi dengan GPS eksternal. Modul arduino kedua merupakan modul utama yang digunakan untuk berkomunikasi dengan modul sekunder melewati protokol I2C dan juga berkomunikasi dengan *flight controller* melalui protocol UART. Skema perakitan modul pemancar lokasi ditunjukan oleh gambar 4.46.



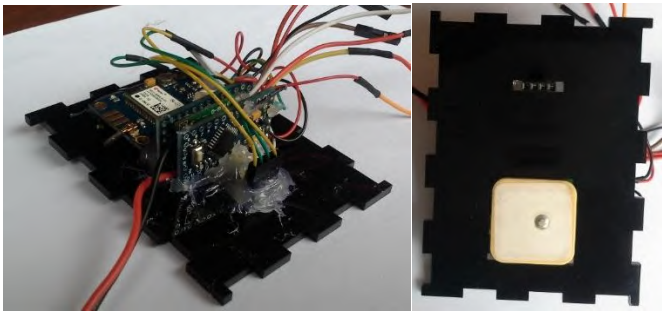
Gambar 4.46 Skema Rangkaian Modul Pemancar Lokasi

Pada gambar 4.46 terlihat adanya LCD display yang digunakan untuk memonitor data-data yang ada pada *flight controller* maupun dari gps eksternal. LCD digunakan untuk mempermudah pengguna mengetahui kondisi GPS eksternal maupun GPS pada quadcopter. Rangkaian module pemancar lokasi ditempatkan pada box akrilik agar terhindar dari geram dan debu seperti pada gambar 4.47.

4.8 Pembuatan Program pada Modul Pemancar Lokasi

Program yang akan dibuat harus mengutamakan keamanan sehingga proses pengiriman data pada *flight controller* harus merupakan data yang valid. Proses pembuatan program dilakukan menggunakan software arduino IDE seperti pada gambar 4.48. Program yang dibuat akan diunggah kedalam module sekunder dan utama. Beberapa library fungsi yang digunakan pada program antara lain:

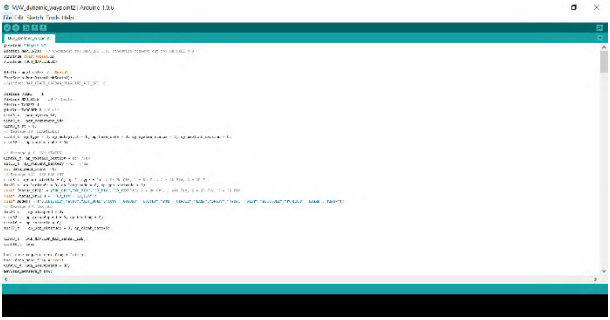
1. GCS_Mavlink.h
2. Fastserial.h
3. U8glib.h



Gambar 4.47 Box Modul Pemancar Lokasi

Library “GCS_Mavlink.h” merupakan protocol untuk mengirim atau menerima packet data dari/ke *flight controller*. Library “Fastserial.h” merupakan fungsi untuk memulai komunikasi serial ke *flight controller* tanpa melewati HAL (*Hardware Abstraction*

Layer) pada software arduino IDE. Library “U8glib.h” digunakan untuk menampilkan data pada layar LCD. File program yang diunggah terdapat pada lampiran.



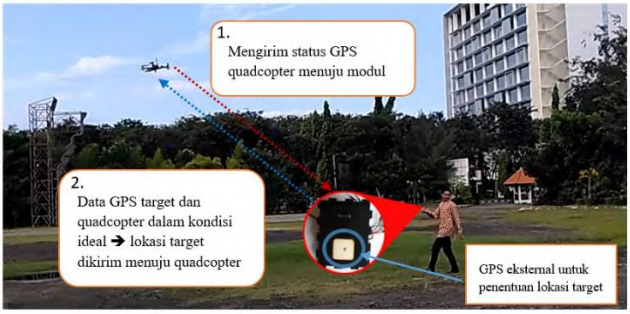
Gambar 4.48 Antar muka software Arduino IDE

4.9 Prinsip Kerja Module Pemancar Lokasi

Prinsip kerja dari modul pemancar lokasi adalah mentransimiskan data lokasi yang valid dari gps eksternal menuju *flight controller*. Module sekunder digunakan untuk mengolah data gps eksternal. Data string pada gps eksternal diolah menjadi beberpa informasi yaitu longitude, lattitude, altitude, jumlah satelit, UTC dan status gps. Data gps akan valid apabila jumlah sinyal satelit yang terkunci oleh modul GPS lebih dari lima. Data valid yang diolah oleh modul sekunder akan dikirim ke modul utama melalui jalur komunikasi I2C.

Modul utama digunakan untuk menerima data dari modul sekunder dan juga menerima data dari *flight controller*. Data yang ditransmisikan oleh *flight controller* ke modul utama adalah lokasi dan juga status GPS pada quadcopter. Urutan komunikasi yang terjadi pada modul pemancar dengan quadcopter ditunjukan pada gambar 4.49. Modul utama akan mengevaluasi status GPS pada quadcopter maupun gps eksternal. Apabila status GPS menunjukan data yang valid, maka modul utama akan mentransmisikan data valid ke *flight controller*. Proses pengiriman data lokasi target dilakukan dengan interval 2 detik sekali. Quadcopter akan

merespon data gps eksternal dan bergerak menuju lokasi target. Apabila sebelum 2 detik quadcopter telah sampai pada lokasi target,maka quadcopter akan menahan posisi pada lokasi target dan menunggu data lokasi baru dari modul pemancar lokasi.



Gambar 4.49 Komunikasi Serial Modul Pemancar Lokasi dengan Quadcopter

4.10 Tahap Pengujian Hasil Perancangan Quadcopter

Data data respon gerak pada quadcopter didapat dari data logger yang terpasang pada *flight controller*. *Flight controller* akan merekam pembacaan sensor ketika quadcopter dinyalakan. Dari hasil pembacaan sensor, didapat data-data yang dibutuhkan untuk mengevaluasi performa quadcopter. dalam pengujian, sangat sulit memberikan sinyal step karena proses pemberian sinyal referensi/setpoint dilakukan ketika quadcopter mengudara. Kriteria yang harus dipenuhi pada pengujian didapat dengan mengamati grafik hubungan sinyal referensi dengan respon aktual pada gerak quadcpter. Respon gerak dikatakan sesuai kriteria ketika respon gerak mengikuti sinyal referensi/setpoint.

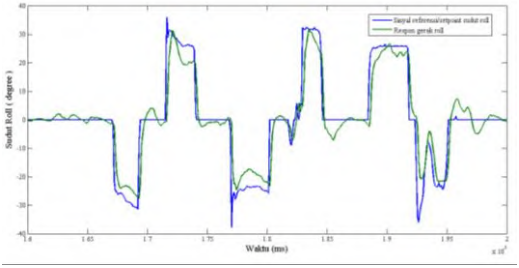
- **Pengujian Gerak Roll**

Pada gambar 4.50 terlihat bahwa respon aktual gerak roll mengikuti sinyal referensi/setpoint. Proses pengujian dilakukan ketika quadcopter dalam kondisi hover. Proses pemberian input referensi/setpoint dilakukan dengan menggunakan remote control

secara manual. Gerak roll aktual mengalami noise karena adanya hambatan angin dan vibrasi. Respon gerak roll yang terlihat pada grafik dapat dikatakan memenuhi kriteria karena selama pengujian gerak quadcopter mengikuti referensi sinyal dari remote control.

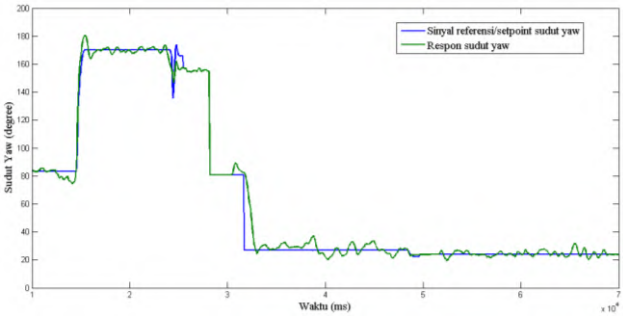
- **Pengujian Gerak Yaw**

Pada gambar 4.51 terlihat respon gerak yaw terhadap sinyal referensi/setpoint. Gerak yaw menglami noise yang signifikan. Proses pemberian sinyal referensi/setpoint dilakukan dengan memberi command melalui komunikasi serial. Mode terbang yang digunakan adalah loiter. *Command* yang diberikan adalah penambahan atau pengurangan sudut *heading* relatif terhadap sudut *heading* quadcopter saat itu.



Gambar 4.50 Grafik Respon Gerak Roll pada Pengujian

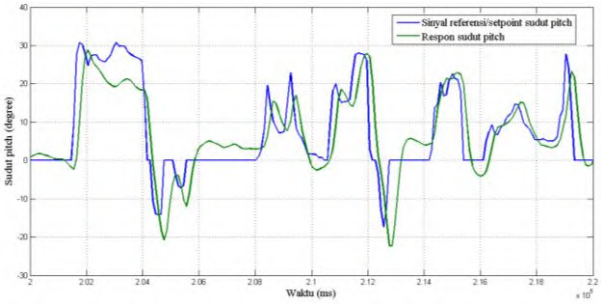
Hasil yang didapat adalah grafik gerak yaw aktual mengalami noise akibat hembusan angin. Diawal grafik terlihat sudut *heading* saat itu adalah 82 derajat, sinyal referensi yang diberikan adalah penambahan 90 derajat. Hasil yang didapat adalah sudut *heading* quadcopter berubah menjadi 172 derajat. Dapat disimpulkan gerak yaw pada rancangan quadcopter sesuai kriteria karena sudut *heading* quadcopter selalu mengikuti sinyal referensi yang diberikan.



Gambar 4.51 Grafik Respon Gerak Yaw pada Pengujian

- **Pengujian Gerak Pitch**

Pada gambar 4.52 terlihat adanya hubungan gerak pitch aktual terhadap sinyal referensi/setpoint. Hasil pengujian yang didapat adalah respon gerak pitch aktual mengikuti referensinya. Proses pemberian sinyal referensi/setpoint dilakukan menggunakan remote control. Adanya *overshoot* tidak menyebabkan quadcopter kehilangan kendali gerak roll karena selama pengujian, respon gerak pitch selalu mengikuti sinyal referensi/setpoint.

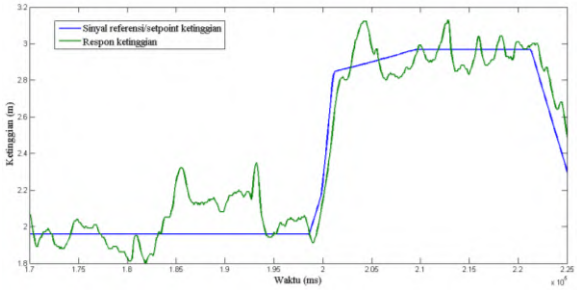


Gambar 4.52 Grafik Respon Gerak Pitch pada Pengujian

- **Pengujian Gerak Vertikal**

Hasil pengujian gerak vertikal terdapat pada gambar 4.53. Pada gambar 4.53 terlihat bahwa perubahan ketinggian pada pembacaan sensor mengikuti referensi/setpoint. Proses pemberian sinyal

referensi/setpoint dilakukan dengan menggunakan remote control. Hasil yang didapat adalah ketinggian quadcopter selalu mengikuti sinyal referensi/setpoint. Pada respon aktual terlihat adanya noise. Noise disebabkan karena adanya gangguan dari hembusan angin dan adanya intensitas cahaya yang mempengaruhi pengukuran sensor barmeter MS5611. Dari hasil pengujian, dapat disimpulkan kendali gerak vertikal pada quadcopter memenuhi kriteria.



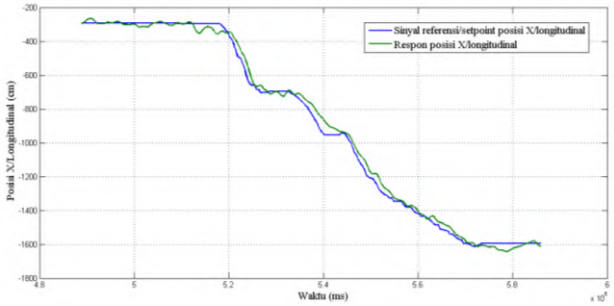
Gambar 4.53 Grafik Respon Ketinggian pada Pengujian

- **Pengujian Gerak Longitudinal**

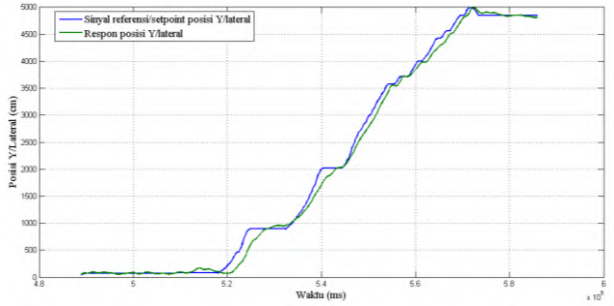
Pengujian gerak longitudinal terlihat pada gambar 4.54. Pada gambar 4.54 terlihat grafik respon gerak longitudinal pada quadcopter mengikuti referensinya. Terdapat noise pada gerak aktual karena radius akurasi gps dalam menentukan lokasi melebar ketika quadcopter tidak dalam keadaan diam.

- **Pengujian Gerak Lateral**

Pengujian gerak lateral terlihat pada gambar 4.55. Pada gambar 4.55 terlihat bahwa gerak lateral pada quadcopter mengikuti referensinya. Adanya penyimpangan gerak terjadi karena radius akurasi gps dalam menentukan lokasi melebar ketika quadcopter tidak dalam keadaan diam.



Gambar 4.54 Grafik Respon Gerak Longitudinal pada Pengujian



Gambar 4.55 Grafik Respon Gerak Lateral pada Pengujian

LAMPIRAN

```
#include "U8glib.h"
#define MAVLINK10
#include <FastSerial.h>
#include <GCS_MAVLink.h>
#define mavlinkSerial  Serial
FastSerialPort0(mavlinkSerial);
#define START    1
#define MSG_RATE    10 // Hertz
#define TARSYS 1
#define TARCOMP 0 //0 all
uint8_t   mav_system_id;
uint8_t   mav_component_id;
uint8_t st = 0;
// Message #0  HEARTHBEAT
uint8_t  ap_type = 0, ap_autopilot = 0, ap_base_mode = 0,
ap_system_status = 0, ap_mavlink_version = 0;
uint32_t  ap_custom_mode = 0;
// Message # 1  SYS_STATUS
uint16_t  ap_voltage_battery = 0;  //mV
int16_t  ap_current_battery = 0;  //dA
int data_send_count = 0;
// Message #24  GPS_RAW_INT
uint8_t  ap_sat_visible = 0, ap_fixtype = 1;  //0= No GPS, 1 = No
Fix, 2 = 2D Fix, 3 = 3D Fix
int32_t  ap_latitude = 0, ap_longitude = 0, ap_gps_altitude = 0;
char* Status_GPS[] =
{"NO_GPS","NO_FIX","2D_FIX","3D_FIX"};// 0 = No GPS, 1
=No Fix, 2 = 2D Fix, 3 = 3D Fix
char* Status_GPS2[] = {"NO_FIX","3D_FIX"};
char* Mode[] =
{"STABILIZE","ACRO","ALT_HOLD","AUTO","GUIDED","L
OITER","RTL","CIRCLE","LAND","DRIFT","SPORT","FLIP",
"AUTOTUNE","POSHOLD","BRAKE","THROW"};
```

```
// Message #74 VFR_HUD
int32_t    ap_airspeed = 0;
uint32_t  ap_groundspeed = 0, ap_heading = 0;
uint16_t  ap_throttle = 0;
int32_t    ap_bar_altitude = 0, ap_climb_rate=0;
unsigned long tracking_timer_start = 0;
unsigned long tracking_timer_stop = 0;
uint8_t   buf[MAVLINK_MAX_PACKET_LEN];
uint16_t  len;
// etc
bool rate_request_sent_flag = false;
bool data_sent_flag = true;
float z = 0;
uint32_t  num_heartbeats = 0U;
mavlink_message_t msg;
#define HeartbeatLed 13
// FUNCTION DEFINITION
uint32_t get_num_heartbeats()
{
  return  num_heartbeats;
}
void do_startup_leds(int n)
  {
    pinMode( HeartbeatLed, OUTPUT);
      // do something to show we are starting up!
      for ( int i = 0; i < n; ++i)
      {
        digitalWrite(HeartbeatLed,HIGH);
        delay(250);
        digitalWrite(HeartbeatLed,LOW);
        delay(250);
      }
  }
void mavlink_setup()  //To run once during setup
{
```

```
  mavlinkSerial.begin(57600);
}
void hb_control()   //To run every loop
{
     mavlink_msg_request_data_stream_pack(mav_system_id,
mav_component_id, &msg, TARSYS, TARCOMP,
MAV_DATA_STREAM_EXTENDED_STATUS, MSG_RATE,
START);
     len = mavlink_msg_to_send_buffer(buf, &msg);
     mavlinkSerial.write(buf,len);
     delay(10);
     mavlink_msg_request_data_stream_pack(mav_system_id,
mav_component_id, &msg, TARSYS, TARCOMP,
MAV_DATA_STREAM_EXTRA2, MSG_RATE, START);
     len = mavlink_msg_to_send_buffer(buf, &msg);
     mavlinkSerial.write(buf,len);
     delay(10);
     mavlink_msg_request_data_stream_pack(mav_system_id,
mav_component_id, &msg, TARSYS, TARCOMP
,MAV_DATA_STREAM_RAW_SENSORS, MSG_RATE,
START);
     len = mavlink_msg_to_send_buffer(buf, &msg);
     mavlinkSerial.write(buf,len);
     rate_request_sent_flag = true;
}
void mavlink_receive()
{
  mavlink_message_t msg;
  mavlink_status_t status;
  while(mavlinkSerial.available())
  {
    uint8_t c = mavlinkSerial.read();

    if(mavlink_parse_char(MAVLINK_COMM_0, c, &msg,
&status))
```

```
   {
     switch(msg.msgid)
     {
       case MAVLINK_MSG_ID_HEARTBEAT:  // 0
       mav_system_id = msg.sysid;
       mav_component_id = msg.compid;
       ap_custom_mode =
mavlink_msg_heartbeat_get_custom_mode(&msg);
       ++num_heartbeats;
         if ( rate_request_sent_flag == false)
             {
               hb_control();
             }
         break;
       case MAVLINK_MSG_ID_GPS_RAW_INT:   // 24
         ap_fixtype =
mavlink_msg_gps_raw_int_get_fix_type(&msg);
// 0 = No GPS, 1 =No Fix, 2 = 2D Fix, 3 = 3D Fix
         ap_sat_visible =
mavlink_msg_gps_raw_int_get_satellites_visible(&msg);          //
numbers of visible satelites
         if(ap_fixtype == 3)
         {
           ap_latitude = mavlink_msg_gps_raw_int_get_lat(&msg);
           ap_longitude =
mavlink_msg_gps_raw_int_get_lon(&msg);
         }
         break;
       case MAVLINK_MSG_ID_VFR_HUD:   //  74
         ap_airspeed = 0;
         ap_groundspeed =
mavlink_msg_vfr_hud_get_groundspeed(&msg);      // 100 =
1m/s
         ap_heading = mavlink_msg_vfr_hud_get_heading(&msg);
// 100 = 100 deg
```

```
        ap_throttle = mavlink_msg_vfr_hud_get_throttle(&msg);
//  100 = 100%
        ap_bar_altitude = mavlink_msg_vfr_hud_get_alt(&msg) *
100;       //  m
        ap_climb_rate=mavlink_msg_vfr_hud_get_climb(&msg) *
100;       //  m/s
        break;
       default:
        break;
     }
    }
  }
}
 void update_leds()
   {
     static unsigned long heartbeat_timer = 0;
     static uint32_t cur_num_heartbeats = 0;
     static bool heartbeat_led_on = false;
     uint32_t num_heartbeats = get_num_heartbeats();
     if ( num_heartbeats > cur_num_heartbeats)
     {
       // new heartbeat
       cur_num_heartbeats = num_heartbeats;
       heartbeat_timer = millis();
       digitalWrite(HeartbeatLed,HIGH);
       heartbeat_led_on = true;
     }else {
       //turn off heartbeat led after a one shot pulse of 1/4 sec
       if ( heartbeat_led_on && (( millis() - heartbeat_timer ) >=
250))
            {
          digitalWrite(HeartbeatLed,LOW);
          heartbeat_led_on = false;
            }
          }
```

```
  }


#define I2C_PULLUPS_ENABLE        PORTC |= 1<<4;
PORTC |= 1<<5;
#define LAT  0
#define LON  1
  int32_t  GPS_coord[2];
  uint8_t  GPS_numSat;
  uint8_t  GPS_update = 0;                    // a binary toogle to
distinct a GPS position update
/**************************************************************
**********************************/
/***************                    I2C GPS
*********************/
/**************************************************************
**********************************/
#define I2C_GPS_ADDRESS             0x20 //7 bits
#define I2C_GPS_STATUS_00            00    //(Read only)
  #define I2C_GPS_STATUS_NEW_DATA       0x01  // New
data is available (after every GGA frame)
  #define I2C_GPS_STATUS_2DFIX          0x02  // 2dfix
achieved
  #define I2C_GPS_STATUS_3DFIX          0x04  // 3dfix
achieved
  #define I2C_GPS_STATUS_NUMSATS        0xF0  // Number
of sats in view
#define I2C_GPS_LOCATION             07    // current location 8
byte (lat, lon) int32_t
uint8_t GPS_NewData(void) {
  uint8_t i2c_gps_status;
  i2c_gps_status =
i2c_readReg(I2C_GPS_ADDRESS,I2C_GPS_STATUS_00);
//Get status register
```

```
  if (i2c_gps_status & I2C_GPS_STATUS_3DFIX) {
//Check is we have a good 3d fix (numsats>5)
    st = 1;// programm LCD
    if (i2c_gps_status & I2C_GPS_STATUS_NEW_DATA) {
//Check about new data
      GPS_numSat = i2c_gps_status >>4;
      i2c_read_reg_to_buf(I2C_GPS_ADDRESS,
I2C_GPS_LOCATION,     (uint8_t*)&GPS_coord[LAT],4);
      i2c_read_reg_to_buf(I2C_GPS_ADDRESS,
I2C_GPS_LOCATION+4,   (uint8_t*)&GPS_coord[LON],4);


      return 1;
    }
  }
  GPS_coord[LAT] = 0;
  GPS_coord[LON] = 0;
  st = 0;
  return 0;
}
int16_t  i2c_errors_count = 0;
#define I2C_SPEED 400000L
static uint8_t rawADC[6];
//
**************************************************
**************************************************
**
// I2C general functions
//
**************************************************
**************************************************
**
void i2c_init(void) {
  //I2C_PULLUPS_ENABLE
```

```
  TWSR = 0;                                  // no prescaler => prescaler =
1
  TWBR = ((F_CPU / 400000) - 16) / 2;        // set the I2C clock
rate to 400kHz
  TWCR = 1<<TWEN;                            // enable twi module, no
interrupt
  i2c_errors_count = 0;
}
void __attribute__ ((noinline)) waitTransmissionI2C(uint8_t
twcr) {
  TWCR = twcr;
  uint8_t count = 255;
  while (!(TWCR & (1<<TWINT))) {
    count--;
    if (count==0) {              //we are in a blocking state => we don't
insist
      TWCR = 0;
      i2c_errors_count++;
      break;
    }
  }
}
void i2c_rep_start(uint8_t address) {
  waitTransmissionI2C((1<<TWINT) | (1<<TWSTA) |
(1<<TWEN)); // send REPEAT START condition and wait until
transmission completed
  TWDR = address;                            // send device
address
  waitTransmissionI2C((1<<TWINT) | (1<<TWEN));             //
wail until transmission completed
}
void i2c_stop(void) {
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);
  //  while(TWCR & (1<<TWSTO));              // <- can produce a
blocking state with some WMP clones
```

```
}
void i2c_write(uint8_t data ) {
  TWDR = data;                          // send data to the previously
addressed device
  waitTransmissionI2C((1<<TWINT) | (1<<TWEN));
}
uint8_t i2c_readAck() {
  waitTransmissionI2C((1<<TWINT) | (1<<TWEN) |
(1<<TWEA));
  return TWDR;
}
uint8_t i2c_readNak() {
  waitTransmissionI2C((1<<TWINT) | (1<<TWEN));
  uint8_t r = TWDR;
  i2c_stop();
  return r;
}

void i2c_read_reg_to_buf(uint8_t add, uint8_t reg, uint8_t *buf,
uint8_t size) {
  i2c_rep_start(add<<1); // I2C write direction
  i2c_write(reg);        // register selection
  i2c_rep_start((add<<1) | 1);  // I2C read direction
  uint8_t *b = buf;
  while (--size) *b++ = i2c_readAck(); // acknowledge all but the
final byte
  *b = i2c_readNak();
}
void i2c_getSixRawADC(uint8_t add, uint8_t reg) {
  i2c_read_reg_to_buf(add, reg, rawADC, 6);
}
void i2c_writeReg(uint8_t add, uint8_t reg, uint8_t val) {
  i2c_rep_start(add<<1); // I2C write direction
  i2c_write(reg);        // register selection
  i2c_write(val);        // value to write in register
```

```
  i2c_stop();
}
uint8_t i2c_readReg(uint8_t add, uint8_t reg) {
  uint8_t val;
  i2c_read_reg_to_buf(add, reg, &val, 1);
  return val;
}
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////
//////////////////////////////////////////////////////LCD
DISPLAY/////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////


U8GLIB_SSD1306_128X64
u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);   // I2C /
TWI
void print_data(void) {
  // graphic commands to redraw the complete screen should be
placed here
  u8g.setFont(u8g_font_u8glib_4);
   u8g.setPrintPos(0, 4);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("GPS_status_AP:");
  u8g.setPrintPos(80, 4);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(Status_GPS[ap_fixtype]);
  u8g.setPrintPos(110, 4);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(ap_sat_visible);
```

```
  u8g.setPrintPos(0, 10);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("LON:");
  u8g.setPrintPos(20, 10);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(ap_longitude);
  u8g.setPrintPos(0, 16);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("LAT:");
  u8g.setPrintPos(20, 16);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(ap_latitude);
  u8g.setPrintPos(0, 22);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("ALT:");
  u8g.setPrintPos(20, 22);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(ap_bar_altitude);
  //--------------------------------//
    u8g.setPrintPos(0, 26);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("--------------------------------");
  u8g.setPrintPos(0, 32);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("MODE :");
  u8g.setPrintPos(50, 32);
  u8g.print(Mode[ap_custom_mode]);
```

```
  u8g.setPrintPos(0, 36);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("--------------------------------");
  //--------------------------------------
   u8g.setPrintPos(0, 42);

  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("GPS_status_target:");
  u8g.setPrintPos(80, 42);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(Status_GPS2[st]);
  u8g.setPrintPos(110, 42);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(GPS_numSat);
  u8g.setPrintPos(0, 50);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("LON:");
  u8g.setPrintPos(20, 50);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(GPS_coord[LON]);
  u8g.setPrintPos(0, 56);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("LAT:");
  u8g.setPrintPos(20, 56);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(GPS_coord[LAT]);
  u8g.setPrintPos(0, 62);
```

```
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print("Location_send_count :");
  u8g.setPrintPos(80, 62);
  // call procedure from base class,
http://arduino.cc/en/Serial/Print
  u8g.print(data_send_count);
}
void send_location_tracking(){
  if((ap_custom_mode == 4)&&(ap_fixtype == 3)&&(st == 1)){
  //--------------------------------------
  if(data_sent_flag == true){
  int32_t alt = ap_bar_altitude;
   z = (float) alt/(float) 100;
   data_sent_flag = false;
   data_send_count++;
  }
  tracking_timer_start = millis();
    if((tracking_timer_start-tracking_timer_stop)>2000){
      tracking_timer_stop = millis();
      int32_t lat = GPS_coord[LAT];
      int32_t lon = GPS_coord[LON];
      float x = (float) lat/(float) 1e7;
      float y = (float) lon/(float) 1e7;

mavlink_msg_mission_item_pack(mav_system_id,mav_compone
nt_id,&msg,TARSYS,TARCOMP,0,MAV_FRAME_GLOBAL,
MAV_CMD_NAV_WAYPOINT,2,0,0,5,0,0,x,y,z);
    len = mavlink_msg_to_send_buffer(buf, &msg);
    mavlinkSerial.write(buf,len);}
    //delay(10);
   //-----------------------------
  }
}
void setup(){
```

```
  i2c_init();
 do_startup_leds(10);
 mavlink_setup();
}
void loop(){
  GPS_NewData();
  mavlink_receive();
  update_leds();
if(ap_custom_mode != 4){
data_sent_flag = true;
}
send_location_tracking();
    u8g.firstPage();
  do {
    print_data();
  } while( u8g.nextPage() );
}
```

BAB 5
PENUTUP

5.1 Kesimpulan

1. Model persamaan fungsi transfer quadcopter dapat didekati dengan model *single input single output*.
2. Hasil simulasi kontrol dengan compensator PD menghasilkan respon sesuai spesifikasi desain. *Overshoot* yang dihasilkan gerak roll sebesar 25.6%, gerak pitch sebesar 29%, gerak yaw sebesar 27.5%, gerak vertikal sebesar 24.3%, gerak lateral sebesar 48% dan gerak longitudinal sebesar 40%. *Settling time* yang dihasilkan oleh gerak roll sebesar 0.689 detik, gerak pitch sebesar 0.565 detik, gerak yaw sebesar 0.611 detik, gerak vertikal sebesar 0.78 detik, gerak longitudinal sebesar 1.4 detik dan gerak lateral sebesar 1 detik.

5.2 Saran

Untuk saran demi menyempurnakan tugas akhir ini adalah:

1. Memodelkan fungsi transfer motor-propeller menggunakan *system identification tool* pada software matlab dengan bantuan arduino dan beberapa perangkat sensor untuk mengukur putaran, *thrust* dan tegangan motor.
2. melakukan proses tuning *compensator* PID guna mendapatkan respon yang lebih baik.
3. Menambah komponen *gimbal stabilizer* kamera untuk kegiatan *aerial photography*.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] Heba, 2014. *Dynamic modelling and Control of a Quadrotor Using Linear and Non-Linear Approaches*. Cairo : The American University in Cairo

[2] Sabatino, Francesco. 2015. *Quadrotor Control: Modeling, nonlinear,control design, and simulation*. Sweden : KTH Electrical Engineering

[3] De Lelis, Marcello. 2011. *Modeling,Identification and Control of a Quadrotor Aircraft*. Prague : Czech Technical University in Prague

[4] Nise, Norman S. 2004. *Control System Engineering.* USA: John Wiley & Sons, Inc.

[5] Brandt, John. 2008. *UIUC Propeller Data Site,* (Online), (http://m-selig.ae.illinois. edu/props/propDB.html).

[6] Ogata, Katsuhiko. 2010. *Modern Control Engineering.* USA: New Jersey

[7] S. Bouabdallah,A. Noth and R.Siegwart.2004. " PID vs LQ control techniques applied to an indoor micro quadrotor. Intelligent Robots and Systems.

[8] Rajesh, Shah M. 2014. "*Mission Planning and Waypoint Navigation of a Micro Quad Copter by Selectable GPS Co-Ordinates*". International Journal of Advance Research in Computer Science and Software Engineering.4,143-152.

[9] Schmidt, Michael David. 2011. *Simulation and Control of Quadrotor Unmanned Aerial Vehicle.* Univeristy of Kentucky Master's theses. 93

BIODATA PENULIS

Dili Kurniawan Putra, adalah anak pertama dari dua bersaudara yang lahir di Sidoarjo, 3 Desember 1991. Penulis telah menempuh pendidikan formal diantaranya SDN Sedati Gede I (1998-2004), SMPN 1 Waru (2004-2007), SMAN 1 Sidoarjo (2007-2010), D3 Teknik Mesin FTI-ITS (2011-2014) dan melanjutkan jenjang S1 Teknik Mesin FTI-ITS (2014-2016) yang kemudian pada tahun 2016 mulai melakukan penelitian di Laboratorium Otomasi Industri sampai dengan terselesaikannya buku ini.

Penulis berusaha untuk berpartisipasi aktif dalam kegiatan yang meningkatkan *softskill* maupun dalam kegiatan akademis selama berkuliah di ITS. Sebelum memasuki jenjang lintas jalur, penulis merupakan anggota aktif laboratorium mekatronika D3 Teknik Mesin ITS. Pengalaman dan wawasan tentang ilmu mekatronika menjadi faktor pendukung terselesaikanya tugas akhir ini. Penulis juga merupakan Wakil Ketua BSO Bengkel D3 Teknik Mesin ITS periode 2013-2014.

E-mail : dilikurniawan11@gmail.com
No. HP : +62 812 9664 1252